LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico - Convocatoria ordinaria de 2026

Instrucciones

- El alumno debe realizar por sí mismo el trabajo. No está permitido copiar los códigos solución de los ejercicios. El trabajo debe realizarse de manera individual. No debe realizarse en grupo. Se penalizará el plagio, así como cualquier uso compartido de las soluciones propuestas y de los códigos programados.
- El trabajo debe entregarse a través del curso virtual de la asignatura.
- La fecha límite de entrega es el día 16 de abril.
- El alumno debe entregar un fichero comprimido, en formato zip o tar, que contenga:
  - Un informe, en formato pdf, en el cual explique la solución a los ejercicios, incluyendo los listados documentados del código C++ desarrollado. Asimismo, en este documento se deben describir algunas de las pruebas realizadas para comprobar que los programas funcionan correctamente y deben mostrarse los resultados obtenidos en dichas ejecuciones de prueba.
  - Los ficheros del código fuente C++ solución a los ejercicios.

  No deben entregarse ficheros ejecutables.

  El nombre del fichero comprimido debe ser la concatenación de los apellidos y el nombre del alumno. Por ejemplo, GomezMartinLuisa.zip

Criterios de evaluación

- Para que el trabajo pueda ser corregido, es imprescindible que el alumno entregue dentro del plazo establecido un fichero comprimido que contenga el informe en formato pdf y el código fuente C++ de los ejercicios que haya realizado.
- Si no entrega el informe, el trabajo se valorará con cero puntos.
- El trabajo se compone de 4 ejercicios, cada uno de los cuales se valorará sobre 2.5 puntos.
- No es obligatorio realizar todos los ejercicios. Para aprobar el trabajo es necesario y suficiente que la nota total obtenida en los ejercicios sea mayor o igual que 5.
- Si el código solución de un ejercicio tiene errores de compilación o no tiene la funcionalidad pedida, dicho ejercicio se valorará con cero puntos.

  Se recomienda comprobar que el programa compila y ejecuta correctamente con el compilador online siguiente:

  `https://www.onlinegdb.com/online_c++_compiler`
- Si el código solución de un ejercicio compila sin errores y tiene la funcionalidad pedida, la puntuación en dicho ejercicio será al menos de 2 puntos.
- Se valorará positivamente la adecuada documentación del código, así como la presentación y calidad de las explicaciones proporcionadas en el informe.

  En el enunciado de los ejercicios se pide que describa en el informe algunas pruebas para comprobar que sus programas funcionan correctamente. Debe mostrar en el informe capturas de pantalla donde se muestren los resultados obtenidos en dichas ejecuciones de prueba.

Ejercicio 1

Un fichero de texto llamado *datosExp.txt* contiene uno o más números reales formando una columna. Sea N el número de números reales que contiene el fichero.

Escriba un programa en C++ que lea este fichero de texto almacenando los números en un vector, transforme dichos números al intervalo $[0, 1]$, y calcule y escriba en la consola (en formato fijo con 4 decimales) qué proporción de estos números transformados se encuentra en cada uno de los diez intervalos siguientes:

$$[0, 0.1),\ [0.1, 0.2),\ [0.2, 0.3),\ [0.3, 0.4),\ \ldots,\ [0.8, 0.9),\ [0.9, 1]$$

La proporción de números contenidos en uno de estos intervalos se calcula dividiendo el número de elementos que hay en el intervalo por el número total de elementos (N).

Supongamos un conjunto de números reales $D = \{d_1, \ldots, d_N\}$. Sean el valor máximo (M) y mínimo (m):

$$M = \text{máx}\,\{d_1, \ldots, d_N\} \qquad\qquad m = \text{mín}\,\{d_1, \ldots, d_N\}$$

Asumiendo que $M > m$, estos números se transforma al intervalo $[0, 1]$ de la forma siguiente:

$$u_i = \frac{d_i - m}{M - m} \qquad\qquad \text{para } i : 1, \ldots, N$$

donde $U = \{u_1, \ldots, u_N\}$ es el conjunto de números obtenido de la transformación. Obsérvese que u_i está en el intervalo $[0, 1]$ para todo $i : 1, \ldots, N$, y que los conjuntos D y U tienen el mismo número N de elementos.

En el caso particular $M = m$, no se realiza la transformación y se asume que la proporción de elementos en los diez intervalos es cero.

Muestre en la memoria capturas de pantalla del resultado obtenido al ejecutar su programa empleando cada uno de los ficheros *datosExp.txt* cuyo contenido es mostrado en los casos de prueba indicados a continuación.

Caso de Prueba 1

Contenido del fichero *datosExp.txt*:

```
3.1
3.1
3.1
```

Caso de Prueba 2

Contenido del fichero *datosExp.txt*:

```
0
1
2
3
4
5
6
7
8
9
10
```

Caso de Prueba 3

Contenido del fichero *datosExp.txt*:

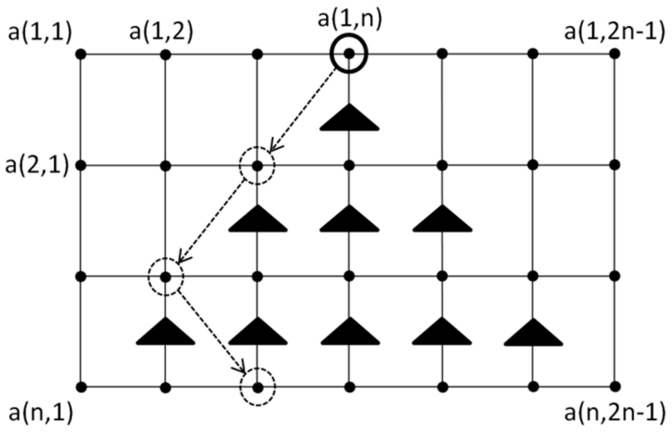
```
-8.0492
-4.4300
 0.9376
 9.1501
 9.2978
-6.8477
 9.4119
 9.1433
-0.2925
 6.0056
-7.1623
-1.5648
 8.3147
 5.8441
 9.1898
 3.1148
-9.2858
 6.9826
 8.6799
 3.5747
 5.1548
 4.8626
-2.1555
 3.1096
-6.5763
 4.1209
-9.3633
-4.4615
-9.0766
-8.0574
 6.4692
 3.8966
-3.6580
 9.0044
-9.3111
-1.2251
-2.3688
 5.3103
 5.9040
-6.2625
-0.2047
-1.0883
 2.9263
```

```
 4.1873
 5.0937
-4.4795
 3.5941
 3.1020
-6.7478
-7.6200
-0.0327
 9.1949
-3.1923
 1.7054
-5.5238
 5.0253
-4.8981
 0.1191
 3.9815
 7.8181
 9.1858
 0.9443
-7.2275
-7.0141
-4.8498
 6.8143
-4.9144
 6.2857
-5.1295
 8.5853
-3.0003
-6.0681
-4.9783
 2.3209
-0.5342
-2.9668
 6.6166
 1.7053
 0.9945
 8.3439
-4.2832
 5.1440
 5.0746
-2.3911
 1.3564
-8.4829
-8.9210
 0.6160
 5.5833
 8.6802
-7.4019
 1.3765
-0.6122
-9.7620
-3.2575
-6.7564
 5.8857
-3.7757
 0.5707
-6.6870
```

Ejercicio 2

Se desea estudiar el comportamiento estadístico de un juego, que consiste en lanzar una bola desde la parte superior de una rampa. El camino de caída desde ese punto inicial se va bifurcando, hasta llegar al final de la rampa. En cada punto de bifurcación de la trayectoria es igualmente probable que la bola caiga en diagonal hacia la izquierda o hacia la derecha.

En la siguiente figura se muestra un ejemplo del comportamiento de la bola. Las posiciones de la bola pueden señalarse en una matriz compuesta por n filas y $2 \cdot n - 1$ columnas. En la figura se muestra el caso $n = 4$.



La bola siempre parte desde la fila 1 y la columna n, es decir, desde la posición $a(1, n)$. Si la bola se encuentra en la posición $a(i, j)$, puede caer con la misma probabilidad en la posición $a(i+1, j+1)$ o en la posición $a(i+1, j-1)$, y así sucesivamente hasta alcanzar la fila n, en la cual la bola se para.

Escriba un programa en C++ que realice N simulaciones del movimiento de caída la bola, usando otras tantas secuencias de números pseudoaleatorios independientes entre sí. Tanto el número de simulaciones N, como el número de filas n, son constantes del programa. Para cada simulación, el programa debe contabilizar en qué columna se encuentra la bola cuando llega al final de la rampa (a la fila n).

Una vez finalizadas las N simulaciones, el programa debe mostrar en la consola el número de veces que la bola ha finalizado su movimiento, quedando parada, en cada una de las columnas.

Además, el programa debe calcular y mostrar en la consola los valores anteriores normalizados respecto al número total de simulaciones N. Es decir, si N_i es el número

de veces que la bola ha finalizado en la columna i, debe mostrarse el valor N_i/N para todo i, escrito en formato científico con 4 dígitos decimales.

Puede emplear la función `int rand (void)`, definida en `<cstdlib>`, para la generación de números pseudoaleatorios.

Casos de prueba. Incluya en la memoria capturas de pantalla donde muestre el resultado de la ejecución de su programa para los valores indicados en cada una de las filas de la tabla siguiente.

Prueba núm.	n	N
1	2	1e4
2	3	1e5
3	4	1e6
4	10	1e8

Ejercicio 3

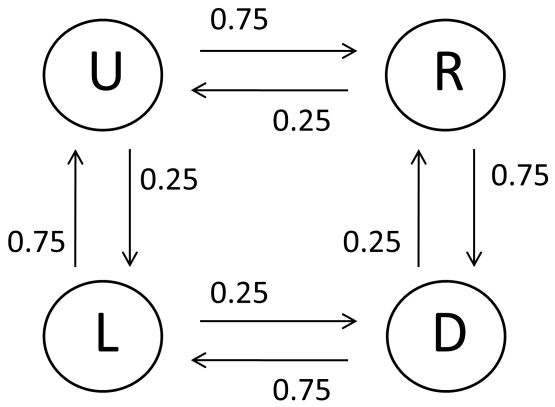
La posición de una partícula se define mediante sus coordenadas cartesianas (x, y). La dirección del eje X corresponde con el movimiento horizontal de la partícula (hacia la izquierda y la derecha) y el eje Y corresponde con el movimiento vertical (hacia arriba o hacia abajo).

La partícula está confinada dentro del área cuadrada $0 \leq x \leq 1000$, $0 \leq y \leq 1000$. La partícula se encuentra inicialmente en el centro del cuadrado, es decir, en el punto $(500, 500)$.

Con cada tic del reloj, la partícula avanza una unidad de longitud hacia arriba (U - up), abajo (D - down), izquierda (L - left) o derecha (R - right). Por tanto, en los instantes correspondientes a los tics del reloj, el valor de las coordenadas x e y de la partícula son números enteros.

El movimiento de la partícula en cada tic del reloj está descrito mediante el diagrama de transición de estados mostrado en la figura. Éste consta de cuatro estados, representados mediante las letras U, R, L y D, que describen la dirección y sentido del movimiento de la partícula: U (hacia arriba: en el sentido Y positivo), D (hacia abajo: en el sentido Y negativo), R (hacia la derecha: en el sentido X positivo) y L (hacia la izquierda: en el sentido X negativo).

Las flechas del diagrama de transición de estados representan las posibles transiciones entre estados. El número escrito junto a cada transición indica la probabilidad de que se produzca dicha transición.



Así, por ejemplo, si la partícula se encuentra en el estado U, existe una probabilidad 0.75 de que pase al estado R y una probabilidad 0.25 de que pase al estado L. Si se

encuentra en el estado R, pasará al estado D con probabilidad 0.75 y al estado U con probabilidad 0.25. Análogamente para las transiciones desde los estados D y L.

Para decidir en cada caso cuál de las dos posibles transiciones se ejecuta, el programa genera un número pseudoaleatorio (puede emplear la función `int rand (void)`, definida en `<cstdlib>`).

El objetivo es simular el movimiento de la partícula, escribiendo en un fichero de texto las coordenadas x e y de su posición en los sucesivos tics del reloj. Cada fila del fichero corresponde con un tic. Deben escribirse los valores de las coordenadas separados por una coma. Por ejemplo, una fila del fichero cuyo contenido sea
`503,200`
indica que la partícula se encuentra en la posición $x = 503$, $y = 200$.

La partícula se encuentra inicialmente en el estado U.

La condición de finalización de la simulación es que la partícula toque alguno de los lados del cuadrado en el que está confinada. Esto sucede cuando la coordenada x vale 0 o 1000, o cuando la coordenada y vale 0 o 1000.

Escriba un programa en C++ que simule el movimiento de la partícula, escribiendo en un fichero de texto llamado *trayectoria.txt* la posición de la partícula en cada uno de los tics del reloj. Además, el programa debe escribir en la consola el número total de tics del reloj que ha durado la simulación. Obsérvese que dicho número debe ser igual al número de filas del fichero *trayectoria.txt*.

Caso de prueba. Ejecute su programa e incluya en la memoria capturas de pantalla que muestren las 10 primeras líneas y las 10 últimas líneas del fichero *trayectoria.txt* generado, así como la escritura en consola del número total de tics del reloj que ha durado la simulación.

Ejercicio 4

Escriba un programa en C++ que realice transformaciones, consistentes en escalados y rotaciones, sobre un array unidimensional de dos componentes x e y de tipo real, el cual es introducido por el usuario a través de la consola. Estas transformaciones se realizan de la forma indicada a continuación.

- **Escalado**. Consiste en realizar el producto del factor de escala, que es un número real a, por el array (x, y). El resultado es el array unidimensional de dos componentes $(a \cdot x, a \cdot y)$.
- **Rotación**. Consiste en realizar el producto de la matriz

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

  por el array (x, y). El ángulo de rotación α pasado como argumento a las funciones trigonométricas debe expresarse en radianes mientras que, como se explica a continuación, el ángulo introducido por el usuario a través de la consola está expresado en grados. Esto implica que el programa debe realizar el cambio de unidades.

También a través de consola, el usuario indica al programa las transformaciones a realizar. Para ello, se emplea el siguiente procedimiento: especificar el factor de escala a continuación de la palabra `ESCALA`, separando ambos por uno o más espacios en blanco, y especificar el ángulo de rotación a continuación de la palabra `ROT_DEG`, separando ambos por uno o más espacios en blanco. Es posible describir varias transformaciones sucesivas, para lo cual deben escribirse una a continuación de otra, separadas por uno o más espacios en blanco. El final de la secuencia de transformaciones se indica mediante el caracter punto y coma (;).

Por ejemplo, la siguiente línea de texto introducida por el usuario a través de la consola

```
ESCALA 1.5 ROT_DEG 30.5 ROT_DEG -10 ESCALA -0.5;
```

o indistintamente (incluyendo más espacios en blanco):

```
ESCALA  1.5 ROT_DEG  30.5  ROT_DEG -10   ESCALA  -0.5 ;
```

indica al programa que debe realizar sobre el array (x, y) una transformación de escala con un factor 1.5, a continuación una rotación de 30.5°, seguidamente una rotación de $-10°$ y finalmente una transformación de escala con factor -0.5.

El programa en C++ debe realizar las acciones descritas a continuación.

1. Mostrar un mensaje en la consola indicando al usuario que introduzca los dos componentes, x e y, del array unidimensional que desea transformar. Leer los valores introducidos por el usuario y almacenarlos en un array unidimensional de dos componentes llamado `v`.

2. Mostrar un mensaje en la consola indicando al usuario que introduzca por consola el texto que describe las transformaciones a realizar sobre el array `v`.

3. Comprobar si la línea de texto leída describe correctamente una o varias transformaciones. En caso afirmativo, realizar las transformaciones sobre el array `v` y mostrar en la consola el resultado obtenido al realizar cada una de las sucesivas transformaciones, escribiendo los componentes del array en formato fijo con tres dígitos decimales. En caso negativo, escribir un mensaje de error en la consola.

4. Terminar.

Por ejemplo, como resultado de las transformaciones
`ESCALA 1.5 ROT_DEG 30.5 ROT_DEG -10 ESCALA -0.5;`
sobre el vector $(1, 2)$, el programa debe escribir en la consola:

```
v = (1.000, 2.000)
Escala a = 1.500, v = (1.500, 3.000)
Rotacion alfa = 0.532 rad, v = (-0.230, 3.346)
Rotacion alfa = -0.175 rad, v = (0.354, 3.335)
Escala a = -0.500, v = (-0.177, -1.668)
```

Muestre en la memoria capturas de pantalla del resultado obtenido al ejecutar su programa para cada uno de los casos de prueba siguientes.

Caso de Prueba 1
Vector: $(4, -3)$
Transformaciones: `ESCALA  2.5  ESCALA -2 ;`

Caso de Prueba 2
Vector: $(4, 2.5)$
Transformaciones: `ROT_DEG -90 ;`

Caso de Prueba 3
Vector: $(1, 2)$
Transformaciones: `ESCALA 1.5 ROT_DEG 30.5 ROT_DEG -10 ESCALA -0.5;`

Caso de Prueba 4
Vector: $(-1, 1)$
Transformaciones: `ROT_DEG 60 ROT_DEG -40 ESCALA 2 ROT_DEG -20 ESCALA 0.5;`