LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico - Convocatoria extraordinaria de 2025

Instrucciones

- El estudiante 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 10 de septiembre.
- 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 la memoria capturas de pantalla donde se muestren los resultados obtenidos en dichas ejecuciones de prueba.

Ejercicio 1

En un fichero de texto llamado *aleatorios.txt* están escritos un número par de números pseudoaleatorios. El contenido del fichero es el siguiente:

```
0.563585 0.193304 0.808741 0.585009 0.479873 0.350291 0.895962 0.822840
0.746605 0.174108 0.858943 0.710501 0.513535 0.303995 0.014985 0.091403
0.364452 0.147313 0.165899 0.988525 0.445692 0.119083 0.004669 0.008911
0.377880 0.531663 0.571184 0.601764 0.607166 0.166234 0.663045 0.450789
0.352123 0.057039 0.607685 0.783319 0.802606 0.519883 0.301950 0.875973
0.726676 0.955901 0.925718 0.539354 0.142338 0.462081 0.235328 0.862239
0.209601 0.779656 0.843654 0.996796 0.999695 0.611499 0.392438 0.266213
0.297281 0.840144 0.023743 0.375866 0.092624 0.677206 0.056215 0.008789
```

Escriba un programa en C++ que genere N observaciones de una distribución normal $\mathrm{N}(\mu, \sigma^2)$. Para ello, el programa debe realizar las acciones siguientes:

1. Solicitar por consola al usuario que introduzca el número de observaciones N que desea obtener. Almacenar el valor leído de consola en una variable entera llamada `N`.

2. Solicitar por consola al usuario que éste introduzca los valores de μ y σ. Almacenar los valores en dos variables llamadas `media` y `stdDev`.

3. Aplicar el *método de Box y Muller* para generar, a partir de los números pseudoaleatorios leídos del fichero, observaciones de la distribución normal N(0,1). Si u_1 y u_2 son dos números pseudoaleatorios, entonces z_1 y z_2 son observaciones independientes de la distribución N(0,1).

$$\begin{aligned} z_1 &= \sqrt{-2 \cdot \ln(u_1)} \cdot \cos(2 \cdot \pi \cdot u_2) \\ z_2 &= \sqrt{-2 \cdot \ln(u_1)} \cdot \sin(2 \cdot \pi \cdot u_2) \end{aligned}$$

4. Transformar las observaciones de la distribución N(0,1) a observaciones de la distribución $\mathrm{N}(\mu, \sigma^2)$. Si z es una observación de la distribución N(0,1), entonces x, calculado de la manera siguiente:

$$x = \mu + \sigma \cdot z$$

   es una observación de la distribución $\mathrm{N}(\mu, \sigma^2)$.

5. Mostrar en la consola las N observaciones de la distribución $\mathrm{N}(\mu, \sigma^2)$. Los números deben escribirse en la consola formando una columna (un número por línea), en formato fijo y con una precisión de 6 dígitos.

6. Terminar.

Caso de prueba. Muestre en la memoria una captura de pantalla de la salida producida por su programa para los valores de entrada $N = 10$, $\mu = 3$, $\sigma = 0.2$.

Ejercicio 2

El *método de Jacobi* es un método iterativo para el cálculo de las soluciones de un sistema de N ecuaciones lineales. Dicho sistema de ecuaciones puede representarse de la forma:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

donde $\mathbf{x}$ es el vector de incógnitas, y la matriz $\mathbf{A}$ y el vector $\mathbf{b}$ están compuestos por números reales conocidos. La matriz $\mathbf{A}$ tiene tamaño $N \times N$. Los vectores $\mathbf{x}$ y $\mathbf{b}$ son vectores columna de N elementos.

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{2,1} & a_{2,2} & \dots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \dots & a_{N,N} \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}$$

Sea x_i el elemento i-ésimo del vector de incógnitas $\mathbf{x}$. El método de Jacobi calcula su valor de manera iterativa, partiendo de un valor inicial. Emplearemos el superíndice k, esto es $x_i^{(k)}$, para representar el valor inicial y los sucesivos valores obtenidos de la iteración para el cálculo de x_i. De esta manera, $x_i^{(0)}$ representa el valor inicial de la iteración y $x_i^{(1)}$, $x_i^{(2)}$, $x_i^{(3)}$, ... representan los sucesivos valores de la iteración, que son calculados aplicando la fórmula siguiente:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \cdot \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{j=N} a_{i,j} \cdot x_j^{(k)} \right) \qquad \text{con } k = 0, 1, \dots$$

Por ejemplo, en el caso $N = 3$, la iteración de los tres elementos del vector de incógnitas se realiza aplicando las tres fórmulas siguientes:

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{1,1}} \cdot \left(b_1 - a_{1,2} \cdot x_2^{(k)} - a_{1,3} \cdot x_3^{(k)} \right) \\ x_2^{(k+1)} &= \frac{1}{a_{2,2}} \cdot \left(b_2 - a_{2,1} \cdot x_1^{(k)} - a_{2,3} \cdot x_3^{(k)} \right) \\ x_3^{(k+1)} &= \frac{1}{a_{3,3}} \cdot \left(b_3 - a_{3,1} \cdot x_1^{(k)} - a_{3,2} \cdot x_2^{(k)} \right) \end{aligned}$$

La condición de parada de la iteración se satisface cuando se ha realizado un determinado número máximo de iteraciones (I), o bien cuando la suma de los valores

absolutos de las diferencias entre el valor anterior y siguiente de las incógnitas sea inferior a un valor umbral U preestablecido, es decir, cuando se satisfaga:

$$U > \sum_{i=1}^{i=N} \left| x_i^{(k+1)} - x_i^{(k)} \right|$$

Escriba un programa en C++ que realice las acciones siguientes:

1. Declarar dos constantes globales llamadas `U` e `I`, y asignarles respectivamente los valores 0.0001 y 100. Estas dos constantes determinan la condición de parada del algoritmo de Jacobi.

2. Escribir un mensaje en la consola solicitando al usuario que introduzca el número de ecuaciones (N) del sistema. Leer el valor introducido, almacenándolo en una variable de tipo entero llamada `N`.

3. Escribir un mensaje en la consola solicitando al usuario que introduzca por consola los elementos de la matriz $\mathbf{A}$ y del vector $\mathbf{b}$. Leer los valores introducidos, almacenando los elementos de $\mathbf{A}$ en un vector de dos dimensiones llamado `A` y los elementos de $\mathbf{b}$ en un vector de una dimensión llamado `b`.

4. Escribir un mensaje en la consola solicitando al usuario que introduzca por consola los valores iniciales para la iteración $x_1^{(0)}, \ldots, x_N^{(0)}$. Leer los valores introducidos y almacenarlos en un vector de una dimensión llamado `xInicial`.

5. Ejecutar el algoritmo de Jacobi hasta que se satisfaga su condición de finalización. Escribir en la consola la solución calculada del sistema de ecuaciones, en formato fijo con 4 dígitos decimales.

6. Terminar.

Caso de prueba. Incluya en la memoria una captura de pantalla del resultado de ejecutar su programa para la matriz $\mathbf{A}$ y el vector $\mathbf{b}$ siguientes:

$$\mathbf{A} = \begin{pmatrix} 31.25 & 3.40 & -0.2 & 1.77 & 10.3 & -13.74 \\ 0.26 & -15.63 & 0.6 & -9.49 & 1.46 & 2.24 \\ 6.59 & 11.26 & 51.47 & -11.27 & 12.59 & 4.29 \\ -9.71 & 9.27 & 20.46 & -95.61 & 34.34 & 11.33 \\ 12.23 & 0.01 & 3.24 & -1.01 & 39.15 & -13.01 \\ 18.28 & -13.29 & -0.01 & 86.34 & 21.23 & 163.05 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -32.3 \\ 2.36 \\ 11.24 \\ 3.98 \\ -1.32 \\ 66.82 \end{pmatrix}$$

escogiendo estos valores iniciales para la iteración: $x_1^{(0)} = x_2^{(0)} = \cdots = x_6^{(0)} = 0$.

Ejercicio 3

Escriba un programa en C++ que aplique las fórmulas de adición de ángulos para seno y coseno a una expresión introducida por el usuario a través de la consola, escribiendo en la consola la expresión resultante. Las fórmulas de adición son:

$$\begin{aligned} \operatorname{sen}(A \pm B) &= \operatorname{sen}(A) \cdot \cos(B) \pm \cos(A) \cdot \operatorname{sen}(B) \\ \cos(A \pm B) &= \cos(A) \cdot \cos(B) \mp \operatorname{sen}(A) \cdot \operatorname{sen}(B) \end{aligned}$$

La expresión introducida por el usuario debe ser el seno o coseno de la suma o resta de dos ángulos. La expresión no debe contener espacios en blanco y los ángulos deben ser números reales mayores o iguales a cero, y estar expresados en formato fijo con uno o más dígitos decimales.

El programa debe comprobar que la expresión introducida por el usuario es correcta, mostrando en la consola el correspondiente mensaje de error si no lo es. Si la expresión introducida es correcta, el programa debe escribir en la consola la expresión equivalente obtenida de aplicar la correspondiente fórmula de adición de los ángulos.

Se muestran a continuación varios ejemplos de ejecución del programa, en los cuales el programa escribe en consola el mensaje:
`Introduzca la expresion:`
el usuario introduce por consola el seno o coseno de la suma o resta de dos ángulos, y el programa escribe en la consola el texto:
`Resultado:`
seguido de la expresión equivalente obtenida de aplicar la correspondiente fórmula de adición de ángulos.

```
Introduzca la expresion:
sen(1.23+0.37)
Resultado: sen(1.23)*cos(0.37)+cos(1.23)*sen(0.37)

Introduzca la expresion:
sen(10.23-0.037)
Resultado: sen(10.23)*cos(0.037)-cos(10.23)*sen(0.037)

Introduzca la expresion:
cos(0.2+0.3724)
Resultado: cos(0.2)*cos(0.3724)-sen(0.2)*sen(0.3724)

Introduzca la expresion:
cos(0.0-0.37243456)
Resultado: cos(0.0)*cos(0.37243456)+sen(0.0)*sen(0.37243456)
```

Los siguientes son algunos ejemplos de expresiones no válidas, que deberían dar como resultado la escritura en consola del mensaje de error:
`-- ERROR: Expresion no valida.`
y la finalización del programa.

Expresión no válida	**Motivo del error**
`sin(2.2+1.4)`	La función no es ni `sen`, ni `cos`
`sen(-0.23+0.37)`	A es negativo
`sen(0.23*0.37)`	El operando no es ni `+`, ni `-`
`cos(0+0.3724)`	A no tiene al menos un dígito decimal
`cos(1.+0.3724)`	A no tiene al menos un dígito decimal
`cos(0.0-4)`	B no tiene al menos un dígito decimal
`cos(0.0-4.)`	B no tiene al menos un dígito decimal
`cos(0.0-4.23e1)`	B no está en formato fijo
`sen[0.23+0.37)`	Apertura del paréntesis
`sen(0.23+0.37}`	Cierre del paréntesis

Como comprobación, el programa debe además evaluar el valor numérico de la expresión introducida por el usuario y de la expresión calculada, escribiendo en la consola el resultado de restar ambas, el cual debe ser un valor muy próximo a cero.

Casos de prueba. Incluya en la memoria capturas de pantalla que muestren el resultado obtenido al ejecutar su programa con cada uno de los ejemplos de entradas descritos en el enunciado: los cuatro de la página anterior y las entradas incorrectas mostradas en esta página.

Ejercicio 4

Consideremos una matriz de tamaño $N \times N$, con $N \geq 1$, tal que cada uno de sus componentes vale 0 ó 1. La matriz se encuentra almacenada en un fichero de texto llamado *matriz.txt*. Cada fila de la matriz es descrita en una línea del fichero. Los componentes de la fila se escriben sin espacios de separación. Por ejemplo,

```
11100
01001
00110
00011
00001
```

describe la matriz $A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$, donde $N = 5$.

La matriz describe los arcos que conectan directamente los nodos de una red. En la red hay N nodos, los cuales están numerados 0, 1, ..., $N-1$. Si el elemento $a_{i,j}$ de la matriz A vale 1, esto indica que existe un arco desde el nodo i hasta el nodo j. Los arcos se recorren en un único sentido. Es decir, si $a_{i,j} = 1$ y $a_{j,i} = 0$, en la red existe un arco desde el nodo i al j, pero no existe un arco desde el nodo j al i.

Escriba un programa en C++ que lea la matriz del fichero *matriz.txt* y que solicite al usuario que introduzca por consola el número de uno de los nodos de la red, al cual denominaremos *nodo origen*. El valor de N (tamaño de la matriz) es conocido una vez se lea el fichero, ya que el programa desconoce a priori el tamaño de la matriz. El nodo origen debe ser un número entero comprendido entre 0 y $N-1$, ambos inclusive.

Al escribir el programa, puede asumir que el fichero *matriz.txt* no contiene errores de formato.

El programa debe calcular y escribir en la consola todos los *nodos alcanzables* desde ese *nodo origen*, es decir, todos los nodos a los que se puede llegar desde el nodo origen transitando a través de los arcos. Para cada nodo alcanzable, el programa debe especificar la longitud del camino desde el nodo origen.

La longitud del camino entre un nodo origen y un nodo destino es un número entero, que es igual al menor número de arcos que hay que transitar para ir desde el nodo origen al nodo destino. Veamos un ejemplo.

Ejemplo. Supongamos que el contenido del fichero de texto *matriz.txt* es:

```
11100
01001
00110
00011
00001
```

A continuación se muestra la salida del programa para 5 ejecuciones diferentes, que corresponden a las 5 elecciones del nodo origen que puede realizar el usuario.

−− Ejecución 1:

```
Nodo origen: 0
Nodos accesibles:
Nodo 0, longitud 0
Nodo 1, longitud 1
Nodo 2, longitud 1
Nodo 3, longitud 2
Nodo 4, longitud 2
```

−− Ejecución 2:

```
Nodo origen: 1
Nodos accesibles:
Nodo 1, longitud 0
Nodo 4, longitud 1
```

−− Ejecución 3:

```
Nodo origen: 2
Nodos accesibles:
Nodo 2, longitud 0
Nodo 3, longitud 1
Nodo 4, longitud 2
```

−− Ejecución 4:

```
Nodo origen: 3
Nodos accesibles:
Nodo 3, longitud 0
Nodo 4, longitud 1
```

−− Ejecución 5:

```
Nodo origen: 4
Nodos accesibles:
Nodo 4, longitud 0
```

Casos de prueba. Muestre en la memoria capturas de pantalla de la salida generada por su programa en los casos de prueba descritos a continuación.

En todos los casos, el contenido del fichero de texto *matriz.txt* es:

```
1111000
0100110
0010000
0001010
0000110
0000011
1000001
```

−− Caso de prueba 1:

```
Nodo origen: 0
```

−− Caso de prueba 2:

```
Nodo origen: 1
```

−− Caso de prueba 3:

```
Nodo origen: 2
```

−− Caso de prueba 4:

```
Nodo origen: 3
```

−− Caso de prueba 5:

```
Nodo origen: 4
```

−− Caso de prueba 6:

```
Nodo origen: 5
```

−− Caso de prueba 7:

```
Nodo origen: 6
```