LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico - Junio de 2021

INSTRUCCIONES

- El trabajo práctico debe realizarse de manera individual. No debe realizarse en grupo. Se penalizará cualquier uso compartido de las soluciones propuestas y de los códigos programados.
- El trabajo debe entregarse a través del curso virtual de la asignatura en la plataforma Alf.
- 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 documento 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 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 dos 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 la memoria en formato pdf y el código fuente C++ de los ejercicios que haya realizado.
- El trabajo se compone de 4 ejercicios, cada uno de los cuales se valorará sobre 2.5 puntos.
- Para aprobar el trabajo es necesario que la suma de las puntuaciones obtenidas 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.
- 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 eficiencia y la adecuada documentación del código, así como la presentación y calidad de las explicaciones proporcionadas en la memoria.

EJERCICIO 1

La *suma de Riemann* es un método para obtener una aproximación de la integral definida I de la función $f(x)$ con respecto a x, en el intervalo finito $[a, b]$, esto es:

$$I = \int_a^b f\,(x) \cdot dx$$

Partiendo del hecho de que la integral definida es igual al área bajo la curva de la función $f(x)$, desde $x = a$ a $x = b$, el método consiste en dividir el área en un conjunto de N rectángulos y sumar sus áreas.

Una de las formas de definir los rectángulos es de manera que la altura del rectángulo sea igual al valor medio de la función en sus extremos. Es decir, si se divide el intervalo $[a, b]$ en N subintervalos $[x_0, x_1]$, $[x_1, x_2]$, $\ldots$, $[x_{N-1}, x_N]$, con $x_0 = a$, $x_N = b$, y $x_i < x_{i+1}$ para $i = 0, \ldots, N-1$, la aproximación a la integral I se calcula de la forma siguiente:

$$I \simeq \sum_{i=0}^{N-1} \frac{f\,(x_{i+1}) + f\,(x_i)}{2} \cdot (x_{i+1} - x_i)$$

Escriba un programa en C++ que calcule la integral definida de la función

$$f(x) = x^m$$

en el intervalo $[a, b]$, donde m es un número entero mayor que cero, y a y b son números reales, satisfaciéndose $a < b$.

El programa debe realizar las acciones siguientes:

1. Declarar una constante global de tipo **int** llamada `N` y asignarle el valor 10000. Esta variable almacena el número N de rectángulos del método.

2. Escribir un mensaje en la consola solicitando al usuario que introduzca los valores de m, a y b. Leer los valores de consola, almacenándolos en una variable **int** llamada `m`, y en variables **double** llamadas `a` y `b`.

3. Comprobar que se satisface $a < b$ y también que se satisface $m > 0$. En caso contrario, el programa debe indicarlo mediante un mensaje escrito en la consola y terminar.

4. Calcular mediante el método descrito anteriormente la integral definida de $f(x) = x^m$ en el intervalo $[a, b]$, dividiendo para ello dicho intervalo en N subintervalos de igual longitud. Esto es:

$$x_{i+1} - x_i = \frac{b-a}{N} \qquad \text{para} \quad i = 0, \ldots, N-1$$

5. Escribir en la consola, en formato científico con 10 dígitos de precisión, el valor estimado de la integral definida, el valor de dicha integral calculado a partir de la expresión analítica de la integral,

$$I = \int_a^b x^m \cdot dx = \left[\frac{x^{m+1}}{m+1}\right]_a^b = \frac{b^{m+1} - a^{m+1}}{m+1}$$

   y también la diferencia entre ambos valores.

6. Terminar.

EJERCICIO 2

Una matriz real $\mathbf{A}$, con N filas y N columnas, es una matriz con diagonal estrictamente dominante por filas cuando satisface:

$$|a_{i,i}| > \sum_{\substack{j=1 \\ j \neq i}}^{N} |a_{i,j}| \qquad \text{para todo} \quad i = 1, \ldots, N$$

donde $|a_{i,j}|$ representa el valor absoluto del elemento de la matriz que está situado en la fila i, columna j.

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

1. Declarar una constante global **int** llamada N y asignarle el valor 4.
2. Declarar un array bidimensional global de **double** llamado $\mathbf{A}$, con tamaño $N \times N$.
3. Solicitar al usuario que introduzca por consola los elementos de $\mathbf{A}$.
4. Calcular si la matriz introducida por el usuario es una matriz con diagonal estrictamente dominante por filas. Mostrar en la consola un mensaje indicándolo. En caso afirmativo, terminar.
5. Buscar si, permutando columnas de la matriz $\mathbf{A}$, es posible obtener una matriz con diagonal estrictamente dominante por filas. Si existe, mostrar en la consola dicha matriz. Si no existe, mostrar un mensaje en la consola indicándolo.
6. Terminar.

A continuación se muestran tres ejemplos: (1) $\mathbf{A}_1$ es una matriz con diagonal estrictamente dominante por filas; (2) $\mathbf{A}_2$ no lo es, pero mediante permutaciones de sus columnas es posible obtener una matriz con diagonal estrictamente dominante por filas: la matriz $\mathbf{A}_1$; (3) $\mathbf{A}_3$ ni es una matriz con diagonal estrictamente dominante por filas, ni puede transformarse en una mediante permutaciones de sus columnas.

$$\mathbf{A}_1 = \begin{pmatrix} 6 & -1 & 2 & 0 \\ 1 & 4 & 1 & -1 \\ 2 & -1 & 18 & 5 \\ 1 & -1 & 0 & -4 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & -1 & 6 & 2 \\ -1 & 4 & 1 & 1 \\ 5 & -1 & 2 & 18 \\ -4 & -1 & 1 & 0 \end{pmatrix}, \quad \mathbf{A}_3 = \begin{pmatrix} 0 & -1 & 6 & 2 \\ -1 & 4 & 1 & 12 \\ 5 & -1 & 2 & 18 \\ -4 & -1 & 1 & 0 \end{pmatrix}$$

EJERCICIO 3

Se desea calcular el valor, en sucesivos instantes de tiempo, del nivel de agua dentro de un depósito cilíndrico que tiene un orificio circular en su base, a través del cual sale el agua.

En el instante inicial, el nivel de agua dentro del depósito es h_0 y se abre el orificio que hay en la base del depósito. Entonces, el agua comienza a salir a través del orificio situado en la base del depósito, disminuyendo el nivel de agua en el depósito a medida que transcurre el tiempo.

El nivel de agua h en el depósito viene descrito por la ecuación diferencial ordinaria siguiente:

$$\frac{dh}{dt} = -\frac{\alpha \cdot A}{B} \cdot \sqrt{2 \cdot g \cdot h}$$

donde t es el tiempo, A es la sección del orificio, B es la sección de la base (circular) del depósito y g es la aceleración gravitatoria. El factor de contracción $\alpha = 0.6$ describe el hecho de que la sección del flujo saliente de agua es menor que la sección del orificio.

Para integrar numéricamente la ecuación diferencial ordinaria anterior, emplearemos el *método de Euler mejorado*. Para explicar el método, representaremos la ecuación de la forma genérica siguiente:

$$\frac{dh}{dt} = f(h)$$

Llamaremos h_i al valor de h en el instante t_i, y h_{i+1} al valor de h en el instante $t_{i+1} = t_i + \Delta t$. El tamaño del paso en el tiempo del método de integración es Δt. El método calcula h_{i+1} a partir de h_i en dos pasos, de la forma descrita a continuación.

1. *Paso predictor*. Se calcula una primera aproximación, $h_{i+1}^{(p)}$, empleando para ello el método de Euler:

$$h_{i+1}^{(p)} = h_i + f(h_i) \cdot \Delta t$$

2. *Paso corrector*. Se calcula h_{i+1} de la forma:

$$h_{i+1} = h_i + \frac{f(h_i) + f(h_{i+1}^{(p)})}{2} \cdot \Delta t$$

Escriba un programa en C++ que calcule y escriba en un fichero de texto llamado *nivelAgua.txt* el valor del nivel de agua h (expresado en cm) en los instantes 0, 10 minutos, 20 minutos, $\ldots$, hasta que el nivel de agua sea menor que h_{fin}.

El fichero debe tener dos columnas: en la primera debe aparecer el instante de tiempo (en minutos) y en la segunda el nivel de agua (en cm) en dicho instante. Los números deben escribirse en formato fijo, con un dígito decimal.

El diámetro de la base circular del depósito mide 200 cm. El diámetro del orificio mide 1 cm. La aceleración gravitatoria vale aproximadamente $g = 980$ cm/s^2. En el instante inicial, $t_0 = 0$ s, el nivel de agua es $h_0 = 250$ cm.

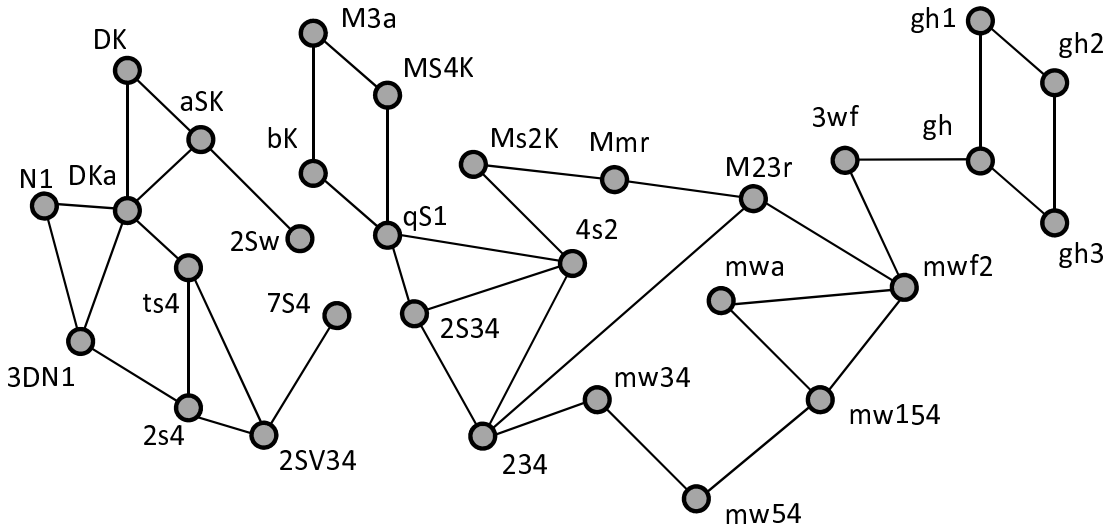
El tamaño del paso del método de integración es $\Delta t = 1$ s. La condición de finalización es que el nivel de agua sea menor que $h_{fin} = 1$ mm.

Además de escribir la información anteriormente indicada en el fichero de texto *nivelAgua.txt*, el programa debe escribir en la consola el instante de tiempo, expresado en segundos, en que se satisface la condición de finalización.

EJERCICIO 4

Un *grafo* consiste en puntos, llamados *vértices*, y líneas conectándolos, llamadas *aristas*. El problema de encontrar el camino más corto a través de una red de ordenadores puede modelarse mediante grafos, en los cuales los vértices representan los ordenadores y las aristas las conexiones entre ellos.

En la figura situada bajo estas líneas se muestra un ejemplo. Cada ordenador tiene asignado un identificador, compuesto por una secuencia de números y/o letras, que lo designa de manera unívoca.



Seleccionados dos vértices, puede no existir ningún camino a través de las aristas que los conecte, puede existir un único camino, o más de uno. La longitud de un camino es igual al número de aristas por las que pasa el camino.

En nuestro problema en concreto, el grafo se almacena en un fichero de texto llamado *redes.txt*, que contiene dos columnas de identificadores. Cada línea del fichero describe una arista, que une los dos vértices indicados mediante los dos identificadores.

Por ejemplo, el fichero *redes.txt* listado a continuación describe el sistema mostrado en la figura anterior.

Fichero redes.txt

```
N1    DKa
N1    3DN1
3DN1  DKa
3DN1  2s4
2s4   ts4
2s4   2SV34
2SV34 ts4
2SV34 7S4
ts4   DKa
DKa   DK
DKa   aSK
DK    aSK
aSK   2Sw
M3a   bK
M3a   MS4K
bK    qS1
MS4K  qS1
qS1   4s2
qS1   2S34
2S34  4s2
2S34  234
234   4s2
234   M23r
4s2   Ms2K
234   mw34
mw34  mw54
mw54  mw154
mw154 mwa
mw154 mwf2
mwa   mwf2
mwf2  M23r
mwf2  3wf
M23r  Mmr
Mmr   Ms2K
3wf   gh
gh    gh1
gh    gh3
gh1   gh2
gh2   gh3
```

Escriba un programa en C++ que realice las acciones descritas a continuación.

1. Leer el fichero *redes.txt*. Si se produce error, indicarlo mediante un mensaje en consola y terminar.
2. Escribir un mensaje en la consola solicitando al usuario que introduzca el identificador del ordenador origen y el identificador del ordenador destino. Leer de consola ambos identificadores.
3. Si alguno de los identificadores introducidos por el usuario no aparece en el fichero *redes.txt*, indicarlo mediante un mensaje escrito en consola y terminar.
4. Determinar si existe al menos un camino entre el ordenador origen y destino y, en caso afirmativo, calcular la longitud del camino más corto. Escribir el resultado en la consola.
5. Terminar.

Seguidamente se muestran tres casos de prueba de la ejecución del programa para el fichero *redes.txt* anteriormente listado.

Caso de prueba 1

```
Introduzca ID origen: N1
Introduzca ID destino: M23r
No existe camino
```

Caso de prueba 2

```
Introduzca ID origen: 7S4
Introduzca ID destino: 2Sw
Longitud camino = 5
```

Caso de prueba 3

```
Introduzca ID origen: M3a
Introduzca ID destino: gh2
Longitud camino = 10
```