LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico - Junio de 2020

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.

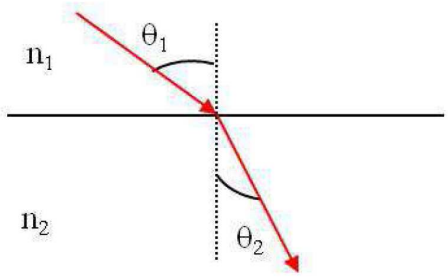
  El nombre del fichero comprimido debe ser la concatenación de los dos apellidos y el nombre del alumno. Por ejemplo, GomezMartinLuisa.zip

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

Cuando un haz de luz incide sobre la superficie de separación entre dos medios, parte de la energía se refleja y parte entra en el segundo medio. El rayo transmitido está contenido en el plano de incidencia del haz, pero cambia de dirección (rayo refractado), formando un ángulo con la normal a la superficie que viene dado por la Ley de Snell,



$$n_1 \cdot sin\,(\theta_1) = n_2 \cdot sin\,(\theta_2)$$

donde n_1 y n_2 son los índices de refracción de los medios 1 y 2, y θ_1 y θ_2 son los ángulos respecto a la normal de los haces incidente y refractado.

Suponemos que el valor del índice de refracción de un medio es un número real, mayor o igual que uno.

Cuando el haz pasa a un medio con menor índice de refracción ($n_1 > n_2$), el ángulo de refracción es mayor que el de incidencia ($\theta_1 < \theta_2$) y, dado que el valor máximo que puede tomar la función seno es uno, hay un ángulo de incidencia límite (θ_C) tal que para ángulos de incidencia mayores la refracción no se produce:

$$n_1 \cdot sin\,(\theta_C) = n_2 \cdot sin\left(\frac{\pi}{2}\right)$$

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

1. Mostrar un mensaje en la consola solicitando al usuario que introduzca por consola los índices de refracción de los dos medios y el ángulo incidente (expresado en radianes).
2. Comprobar si los datos introducidos por el usuario son válidos. Los índices de refracción deben ser mayores o iguales a uno, y el ángulo incidente debe estár en el intervalo $[0, \frac{\pi}{2})$. Si alguna de estas condiciones no se cumple, mostrar un mensaje en consola indicándolo y terminar.
3. Mostrar en la consola, o bien el valor del ángulo de refracción (en radianes), o bien un mensaje indicando que no se produce refracción.
4. Terminar.

EJERCICIO 2

Dados un conjunto de puntos y una recta en el plano, puede calcularse la distancia entre cada punto y la recta, así como el valor máximo de dichas distancias. Finalmente, puede obtenerse qué puntos del conjunto se encuentran a la distancia máxima calculada.

Las coordenadas cartesianas de los puntos están escritas en un fichero de texto llamado *puntos.txt*, que contiene dos columnas de números. Cada fila describe la posición de un punto: en la primera columna está su coordenada X y en la segunda su coordenada Y. Un mismo punto puede aparecer varias veces en fichero.

Los tres parámetros (a, b, c) que determinan la recta serán introducidos por el usuario del programa a través de la consola.

El programa deberá escribir los resultados en la consola.

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

1. Escribir un mensaje en la consola solicitando al usuario que introduzca por consola tres números, que llamaremos a, b y c, a fin de especificar una recta en el plano de la forma:

   $$a \cdot x + b \cdot y + c = 0$$

   Leer los valores introducidos por el usuario. Si $a = b = 0$, mostrar un mensaje en consola indicándolo y terminar.

2. Ir leyendo del fichero *puntos.txt* las coordenadas de los puntos. Al leer cada punto, el programa debe calcular la distancia de ese punto a la recta y sólo almacenar dicha distancia y la coordenada del punto en caso de que la distancia sea la mayor calculada hasta ese momento.

   Si la lectura del fichero produce error, mostrar un mensaje en la consola indicándolo y terminar.

3. Mostrar en la consola el valor de la distancia máxima, así como las coordenadas de los puntos que se encuentran a esa distancia de la recta.

   Aunque el fichero de entrada puede contener puntos repetidos, la salida del programa no debe contener un mismo punto varias veces.

4. Terminar.

EJERCICIO 3

Consideremos la siguiente ecuación de ondas

$$u^2 \cdot \frac{\partial^2 \Phi}{\partial x^2} = \frac{\partial^2 \Phi}{\partial t^2}$$

donde u es la velocidad de la onda y $\Phi(x,t)$ es una función dependiente de la coordenada espacial x y del tiempo t. Para resolver numéricamente dicha ecuación, empleamos la discretización en diferencias finitas siguiente:

$$u^2 \cdot \frac{\Phi(i+1,j) - 2 \cdot \Phi(i,j) + \Phi(i-1,j)}{(\Delta x)^2} = \frac{\Phi(i,j+1) - 2 \cdot \Phi(i,j) + \Phi(i,j-1)}{(\Delta t)^2}$$

donde $\Phi(i,j)$ es una aproximación de $\Phi(x,t)$, con

$$\begin{aligned} x &= i \cdot \Delta x \\ t &= j \cdot \Delta t \end{aligned}$$

para $i, j = 0, 1, 2, \ldots$

Manipulando la ecuación en diferencias anterior, se obtiene la siguiente fórmula explícita para la ecuación de ondas:

$$\Phi(i,j+1) = 2 \cdot (1-r) \cdot \Phi(i,j) + r \cdot (\Phi(i+1,j) + \Phi(i-1,j)) - \Phi(i,j-1)$$

donde la constante r, denominada *relación de aspecto*, se define como:

$$r = \left(\frac{u \cdot \Delta t}{\Delta x} \right)^2$$

Para que el método sea estable, debe escogerse un valor $r \leq 1$.

Obsérvese que, para calcular Φ en el instante $j+1$, es necesario conocer Φ en los dos instantes previos j y $(j-1)$.

Escriba un programa en C++ que, empleando la discretización explícita descrita anteriormente, resuelva la ecuación de ondas

$$u^2 \cdot \frac{\partial^2 \Phi}{\partial x^2} = \frac{\partial^2 \Phi}{\partial t^2}, \qquad 0 < x < 1, \qquad t \geq 0$$

con las condiciones de contorno

$$\Phi(0,t) = 0 = \Phi(1,t), \qquad\qquad t \geq 0$$

y las condiciones iniciales

$$\begin{aligned} \Phi(x,0) &= \sin(\pi \cdot x), \qquad 0 < x < 1, \\ \left.\frac{\partial \Phi}{\partial t}\right|_{t=0} &= 0, \qquad 0 < x < 1 \end{aligned}$$

La velocidad de la onda (u), el tamaño del paso en el tiempo (Δt) y el tamaño del paso en la coordenada espacial (Δx) deben ser constantes del programa, con los valores siguientes: $u = 1$, $\Delta x = 0.1$ y $\Delta t = 0.1$.

El programa debe escribir en un fichero de texto llamado *resultados.txt* la solución $\Phi(i,j)$ obtenida en las posiciones $x = 0, 0.1, 0.2, \ldots, 1$ en cada uno de los instantes de tiempo $t = 0, 0.1, 0.2, \ldots, 1$. El fichero debe contener 11 filas (una para cada instante de tiempo, $j = 0, 1, \ldots, 10$) y 11 columnas (una para cada posición, $i = 0, 1, \ldots, 10$). La solución $\Phi(i,j)$ debe escribirse en formato fijo, con 4 dígitos de precisión.

Obsérvese que para calcular Φ en $j = 1$ es necesario conocer Φ en $j = 0$ y en $j = -1$. Con este propósito, la condición inicial

$$\left.\frac{\partial \Phi}{\partial t}\right|_{t=0} = 0$$

puede discretizarse de la forma

$$\left.\frac{\partial \Phi(x,0)}{\partial t}\right|_{t=0} \simeq \frac{\Phi(i,1) - \Phi(i,-1)}{2 \cdot \Delta t} = 0$$

de donde

$$\Phi(i,1) = \Phi(i,-1)$$

Sustituyendo en la fórmula explícita de la ecuación de ondas, se obtiene

$$\Phi(i,1) = 2 \cdot (1-r) \cdot \Phi(i,0) + r \cdot (\Phi(i+1,0) + \Phi(i-1,0)) - \Phi(i,1)$$

de donde

$$\Phi(i,1) = (1-r) \cdot \Phi(i,0) + \frac{r}{2} \cdot (\Phi(i+1,0) + \Phi(i-1,0))$$

EJERCICIO 4

Cuando hay que evaluar repetidamente un polinomio

$$p_n(x) = \sum_{i=0}^{n} a_i \cdot x^i = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n$$

para diferentes valores de x, es deseable realizar dicha evaluación eficientemente, particularmente cuando el valor de n es grande. A continuación se muestra un procedimiento para ello, basado en la observación de que si el polinomio $p_n(x)$ se expresa en su forma anidada,

$$p_n(x) = a_0 + x \cdot (a_1 + x \cdot (a_2 + \cdots + x \cdot (a_{n-1} + x \cdot a_n) \cdots))$$

basta con realizar n multiplicaciones y n sumas para evaluarlo en $x = x_0$:

$$\begin{aligned} b_n &= a_n \\ b_i &= b_{i+1} \cdot x_0 + a_i, \qquad\qquad i = n-1, \ldots, 0 \end{aligned}$$

satisfaciéndose que $b_0 = p_n(x_0)$.

Veamos un ejemplo. Para $n = 4$, el polinomio escrito de forma anidada sería

$$p_4(x) = a_0 + x \cdot (a_1 + x \cdot (a_2 + x \cdot (a_3 + x \cdot a_4)))$$

El valor del polinomio en $x = x_0$ puede calcularse mediante la secuencia siguiente de operaciones:

$$\begin{aligned} b_4 &= a_4 \\ b_i &= b_{i+1} \cdot x_0 + a_i, \qquad\qquad i = 3, \ldots, 0 \end{aligned}$$

Es sencillo comprobar que el valor b_0 obtenido es igual que $p_4(x_0)$. A continuación se realiza dicha comprobación:

$$\begin{aligned} b_4 &= a_4 \\ b_3 &= b_4 \cdot x_0 + a_3 = a_4 \cdot x_0 + a_3 \\ b_2 &= b_3 \cdot x_0 + a_2 = (a_4 \cdot x_0 + a_3) \cdot x_0 + a_2 \\ b_1 &= b_2 \cdot x_0 + a_1 = ((a_4 \cdot x_0 + a_3) \cdot x_0 + a_2) \cdot x_0 + a_1 \\ b_0 &= b_1 \cdot x_0 + a_0 = (((a_4 \cdot x_0 + a_3) \cdot x_0 + a_2) \cdot x_0 + a_1) \cdot x_0 + a_0 \end{aligned}$$

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

1. Escribiendo un mensaje en la consola, solicitar al usuario que éste introduzca por consola un polinomio cualquiera de coeficientes enteros.
2. Leer la cadena de caracteres introducida por el usuario, almacenándola en un string.
3. Comprobar que la cadena de caracteres leída tiene la estructura descrita a continuación. En caso contrario, indicarlo mediante un mensaje en la consola y terminar.

   La cadena de caracteres debe estar formada por la concatenación de una o más subcadenas. El contenido de cada una de estas subcadenas, al ser leída de izquierda a derecha, debe ser el siguiente:

   – Comenzar con el carácter '$+$' o el carácter '$-$'.
   – A continuación, cero, uno o más dígitos enteros ('0', '1', . . . , '9').
   – Seguidamente, el carácter 'x'.
   – Finalmente, cero, uno o más dígitos enteros ('0', '1', . . . , '9').
4. Calcular a partir de la cadena de caracteres y escribir en la consola el valor de los coeficientes a_0, a_1, . . . , a_n del polinomio que son distintos de cero. Si todos con cero, escribir un mensaje indicándolo y terminar.
5. Escribir en la consola el resultado de evaluar el polinomio en $x = 1.5$, empleando para ello el método descrito en el enunciado del ejercicio.
6. Terminar.

A continuación se muestran tres ejemplos de entrada válida al programa.

```
+1-x+4x2-16x5
+4x2+1-16x5-x
-16x5-x2+4+7x2-3-x-2x2
```

Cualquiera de las tres debería producir la salida siguiente:

```
a0 = 1
a1 = -1
a2 = 4
a5 = -16
p(1.5) = -113
```