LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico - Septiembre 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 10 de septiembre.
- El alumno debe entregar un fichero comprimido, en formato zip o tar, que contenga:
  - Una *memoria* en la cual explique la solución a los ejercicios, incluyendo los listados documentados del código C++ desarrollado y una descripción detallada de las pruebas realizadas al código para comprobar que funciona correctamente. Este documento deberá estar en formato pdf.
  - 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 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 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.
- 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.

Contenido de la memoria

Debe incluirse en la memoria, para cada uno de los ejercicios:

1. Listado del código fuente debidamente documentado.
2. Enumeración del conjunto completo de pruebas que usted ha realizado al código para comprobar que funciona correctamente.
3. Un ejemplo de ejecución del código, incluyendo capturas de pantalla en las que puedan verse las distintas fases en la ejecución del programa, tales como la entrada de los datos, la salida de los resultados, la escritura en la consola de mensajes de error, etc. según proceda en cada caso.

Ejercicio 1

Un método numérico para calcular un valor de z que satisface la ecuación $P(z) = 0$ es el método de Newton. Partiendo de un valor aproximado z_0 de la solución, se van calculando sucesivas aproximaciones (z_1, z_2, ...) a la solución mediante la fórmula iterativa (1.1), donde $P'(z)$ representa la derivada de $P(z)$ respecto a z.

$$z_{n+1} = z_n - \frac{P(z_n)}{P'(z_n)} \qquad \text{para} \quad n = 0, 1, 2, \ldots \tag{1.1}$$

La iteración finaliza cuando se satisface al menos una de estas dos condiciones:

1. El módulo de $P(z_{n+1})$ está suficientemente próximo a cero, es decir, se satisface $|P(z_{n+1})| < \varepsilon$, siendo ε un cierto valor real predefinido.
2. El número de iteraciones ha superado un cierto valor máximo N, siendo N un número entero predefinido.

Escriba un programa en C++ que emplee el método de Newton para estimar las raíces complejas del polinomio:

$$z^3 - 1 = 0$$

Los parámetros ε y N deben declararse como constantes globales. El programa debe solicitar al usuario que éste introduzca por consola las partes real e imaginaria de z_0. A continuación, el programa debe escribir en la consola, en formato científico, con 10 dígitos decimales, los sucesivos valores

$$\begin{aligned} &z_1, \quad |P(z_1)| \\ &z_2, \quad |P(z_2)| \\ &\ldots \end{aligned}$$

obtenidos de aplicar el método de Newton. Finalmente, el programa debe escribir un mensaje en la consola indicando cuál ha sido la condición de finalización: solución encontrada o límite máximo de iteraciones excedido.

Muestre en la memoria el resultado de ejecutar el programa para cada uno de los cuatro conjuntos de valores siguientes:

$\{\varepsilon = 10^{-5}, N = 10, z_0 = 2 + 2i\}$ $\qquad$ $\{\varepsilon = 10^{-5}, N = 10, z_0 = -2 - 2i\}$

$\{\varepsilon = 10^{-5}, N = 10, z_0 = -2 + 2i\}$ $\qquad$ $\{\varepsilon = 10^{-10}, N = 5, z_0 = -10i\}$

Ejercicio 2

Un perro y un pato nadan dentro de un estanque que tiene forma circular, con radio R. El **pato** nada siempre pegado a la orilla del estanque, con velocidad constante, realizando siempre el giro alrededor del estanque en sentido antihorario. El tiempo que tarda el pato en dar una vuelta completa al estanque es T. Empleando un sistema de coordenadas cartesiano X-Y cuyo origen está en el centro del estanque, la posición (x, y) del pato en el instante de tiempo t es:

$$\left(R \cdot \cos\left(\frac{2 \cdot \pi}{T} \cdot t \right),\ R \cdot \text{sen}\left(\frac{2 \cdot \pi}{T} \cdot t \right) \right)$$

El **perro** se encuentra inicialmente en el centro del estanque e intenta acercarse al pato moviéndose de la manera descrita a continuación. En el instante inicial ($t = 0$), el perro levanta la cabeza y observa la posición del pato. Entonces baja la cabeza y nada con velocidad v constante, durante un tiempo τ, en línea recta hacia el punto donde ha visto en $t = 0$ que se encuentra el pato. En el instante $t = \tau$ el perro levanta nuevamente la cabeza y observa la posición del pato en ese instante. Entonces baja la cabeza y, cambiando su dirección de movimiento, nada con velocidad v constante, durante un tiempo τ, en línea recta hacia el punto donde ha visto en $t = \tau$ que se encuentra el pato. En el instante $t = 2\tau$ el perro levanta nuevamente la cabeza, observa la posición del pato y cambia nuevamente su dirección de movimiento. Así sucesivamente.

Escriba un programa en C++ que declare los parámetros R, T, v y τ como constantes globales del programa, y que calcule y escriba en la consola, en los instantes de tiempo $t = 0$, τ, 2τ, 3τ, ..., los valores siguientes:

1. Las coordenadas X-Y de la posición del perro.
2. Las coordenadas X-Y de la posición del pato.
3. La distancia entre el centro del estanque y la posición del perro.
4. La distancia entre el perro y el pato.

La condición de finalización del programa es que el perro salga del estanque o que el pato de una vuelta completa al estanque. Muestre en la memoria el resultado de la ejecución del programa para el conjunto siguiente de valores de los parámetros:

$$R = 20 \text{ m} \qquad T = 200 \text{ s} \qquad v = 0.5 \text{ m/s} \qquad \tau = 2 \text{ s}$$

Ejercicio 3

En un fichero de texto llamado *puntos.txt* se encuentran escritas las coordenadas cartesianas en el plano de N puntos diferentes, con $N \geq 3$. Cada punto es descrito en una fila del fichero: en la primera columna se encuentra el valor de su coordenada X y en la segunda el valor de su coordenada Y. El fichero contiene N filas (no hay puntos repetidos).

Por ejemplo, si el contenido del fichero es el mostrado debajo de estas líneas, a la izquierda, éste especifica los $N = 4$ puntos en el plano indicados a la derecha.

```
0.3   1.7
2.6  -1.4
1.5   1.5
0   -11.2
```

$$\begin{aligned} P_1 &= (0.3, 1.7) \\ P_2 &= (2.6, -1.4) \\ P_3 &= (1.5, 1.5) \\ P_4 &= (0, -11.2) \end{aligned}$$

Como puede verse en el ejemplo anterior, los puntos son nombrados numerándolos consecutivamente, de acuerdo al orden en que aparecen en el fichero. Así, los cuatro puntos definidos en el fichero son nombrados P_1, P_2, P_3 y P_4.

Escogiendo 3 de los N puntos anteriores, puede calcularse el área del triángulo que tiene como vértices dichos puntos. Si los tres puntos están alineados, consideramos que dicho área vale cero.

Escriba un programa en C++ que calcule el área de cada uno de los posibles triángulos que pueden formarse con los N puntos del fichero, y escriba en la consola dichos triángulos y sus áreas, ordenados crecientemente de acuerdo al valor del área.

En la salida del programa por consola, la forma de designar un triángulo es escribiendo, separados por coma, los puntos que definen sus vértices. Así, por ejemplo, la salida por consola correspondiente al fichero anterior debería ser:

```
P1,P2,P3: 1.63
P1,P3,P4: 7.77
P2,P3,P4: 9.16
P1,P2,P4: 15.3
```

Obsérvese que la ordenación de los vértices es indiferente a la hora de especificar el triángulo. Por ejemplo, las siguientes:

`P1,P2,P3`	`P2,P1,P3`	`P3,P1,P2`
`P1,P3,P2`	`P2,P3,P1`	`P3,P2,P1`

son formas equivalentes de designar el mismo triángulo, por lo cual el programa sólo debe escribir una cualquiera de ellas en la consola.

Al realizar el programa, puede asumir que el formato del fichero *puntos.txt* es correcto.

Muestre en la memoria el resultado obtenido de ejecutar su programa para el fichero *puntos.txt* siguiente:

```
-2.4     0.8
-6.2    10.4
 9.987 -11.2
 0       0
 1       1
 2       2
13.4    -3.8
41.2   -18.4
```

Ejercicio 4

Consideremos la Sección de Control de Calidad de una fabrica de piezas metálicas. En dicha sección se realiza una inspección visual del acabado de la pieza y una medida de resistencia mecánica. Todas las piezas son sometidas a estas dos pruebas (en cualquier orden).

Los posibles resultados de la prueba de inspección visual son: OK (la pieza supera la prueba), REWORK (la pieza debe ser retrabajada) y SCRAP (la pieza debe ser desechada). El resultado de la prueba de resistencia mecánica es un valor numérico entero comprendido entre 0 y 5.

Las operaciones realizadas en la Sección de Control de Calidad son registradas automáticamente en un fichero de texto llamado *trazaActividad.txt*, tal como se describe a continuación.

Cada vez que se recibe una pieza en la Sección de Control de Calidad, se añade una línea al final del fichero con el formato siguiente:

```
ARRIVE <ID> <PRIORITY>
```

donde `<ID>` es un número entero que identifica la pieza de manera unívoca a través de todo el proceso de control de calidad y `<PRIORITY>` es una etiqueta que indica la prioridad. Esta última puede tomar dos posibles valores: URGENT (la pieza debe ser probada lo antes posible) y NORMAL.

Al finalizar la inspección visual de una pieza, se añade una línea al final del fichero con el formato siguiente:

```
VISUAL <ID> <PROC_TIME> <RESULT_VISUAL>
```

donde `<PROC_TIME>` es un valor real que indica el tiempo empleado en inspeccionar visualmente la pieza y `<RESULT_VISUAL>` indica el resultado, pudiendo tomar tres posibles valores: OK, REWORK y SCRAP.

Al finalizar la prueba mecánica de una pieza, se añade una línea al final del fichero con el formato siguiente:

```
RESIST <ID> <PROC_TIME> <VALUE_RESIST>
```

donde `<PROC_TIME>` es un valor real que indica el tiempo empleado en probar mecánicamente la pieza y `<VALUE_RESIST>` es el resultado obtenido, pudiendo ser un número entero del conjunto $\{0, 1, 2, 3, 4, 5\}$.

Escriba un programa en C++ que lea el contenido del fichero *trazaActividad.txt* y escriba en la consola el resultado de realizar los cálculos siguientes:

- Tiempo medio de proceso en la inspección visual de las piezas con prioridad URGENT.
- Tiempo medio de proceso en la prueba mecánica de las piezas con prioridad URGENT.
- Número de piezas con prioridad URGENT cuyo resultado en la inspección visual es OK y además obtienen en el test mecánico un valor mayor que 2.
- Número de piezas (con cualquier prioridad) cuyo resultado en la inspección visual es REWORK o SCRAP.

Al realizar el programa, puede asumir que el formato del fichero *trazaActividad.txt* no contiene errores. Muestre en la memoria el resultado obtenido al ejecutar su programa con el fichero *trazaActividad.txt* siguiente:

```
ARRIVE 1 NORMAL
VISUAL 1 12.3 OK
ARRIVE 232 URGENT
ARRIVE 102 URGENT
VISUAL 232 15.1 OK
RESIST 1 87.4 2
RESIST 232 68.3 4
ARRIVE 2 NORMAL
RESIST 2 103.8 3
VISUAL 102 6.3 OK
ARRIVE 13 NORMAL
RESIST 102 99.87 5
VISUAL 13 8.4 SCRAP
VISUAL 2 22.3 REWORK
ARRIVE 4 NORMAL
RESIST 13 87.64 4
VISUAL 4 22.4 SCRAP
ARRIVE 122 URGENT
RESIST 4 77.8 1
VISUAL 122 18.4 REWORK
RESIST 122 109.4 2
ARRIVE 532 URGENT
ARRIVE 5 NORMAL
VISUAL 532 16.6 OK
RESIST 532 73.14 4
ARRIVE 132 URGENT
VISUAL 5 13.13 SCRAP
VISUAL 132 11.6 OK
RESIST 132 69.4 5
RESIST 5 91.4 1
```