LENGUAJES DE PROGRAMACIÓN

Trabajo Práctico - Convocatoria ordinaria de 2023

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 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.
- 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. Recuerde que en el informe debe describir algunas de las pruebas que ha realizado para comprobar que sus programas funcionan correctamente y que debe mostrar los resultados obtenidos en dichas ejecuciones de prueba.

Ejercicio 1

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

1. Mediante un mensaje escrito en la consola, solicitar al usuario que introduzca por consola un valor de la temperatura, expresada en una cualquiera de estas tres unidades: grados Celsius, grados Fahrenheit o Kelvin.

   El formato en el cual el usuario debe escribir la temperatura en la consola es el siguiente: debe escribir un número, seguido de uno o más espacios en blanco, a continuación una letra de entre el conjunto {C, F, K}, la cual representa las unidades, y a continuación pulsar la tecla *enter*.

2. Leer el valor introducido por consola. Calcular el valor de la temperatura expresada en las otras dos unidades. Las tres unidades de temperatura se relacionan de la forma siguiente:

$$^{o}\mathrm{F} = ^{o}\mathrm{C} \cdot 1.8 + 32 \qquad\qquad ^{o}\mathrm{C} + 273.15 = \mathrm{K}$$

3. Escribir en la consola la temperatura expresada en cada una de las tres unidades (^{o}C, ^{o}F y K), en formato fijo con 2 dígitos decimales.

4. Terminar.

Al escribir el programa, usted puede asumir que el texto introducido por el usuario tiene el formato correcto.

A continuación se muestra un ejemplo de ejecución del programa.

```
Escriba temperatura con unidades: 25.4 C
25.40 C, 77.72 F, 298.55 K
```

En este ejemplo, primeramente el programa escribe en la consola:
`Escriba temperatura con unidades:`
A continuación, el usuario escribe en la consola: `25.4 C`
El programa calcula la equivalencia de esa temperatura en ^{o}F y K, y escribe en la consola la temperatura expresada en las tres unidades:
`25.40 C, 77.72 F, 298.55 K`

Ejercicio 2

El número complejo $a + bi$ puede representarse mediante la matriz

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

La suma y el producto de dos números complejos puede calcularse sumando y multiplicando, respectivamente, sus matrices.

Por ejemplo, la matriz $M = \begin{pmatrix} 5 & -7 \\ 7 & 5 \end{pmatrix}$ representa el número complejo $z_1 = 5 + 7i$ y la matriz $N = \begin{pmatrix} 1 & 8 \\ -8 & 1 \end{pmatrix}$ representa el número complejo $z_2 = 1 - 8i$.

La suma de las matrices, $\begin{pmatrix} 5 & -7 \\ 7 & 5 \end{pmatrix} + \begin{pmatrix} 1 & 8 \\ -8 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 1 \\ -1 & 6 \end{pmatrix}$, representa $z_1 + z_2 = 6 - i$.

El producto de las matrices, $\begin{pmatrix} 5 & -7 \\ 7 & 5 \end{pmatrix} \cdot \begin{pmatrix} 1 & 8 \\ -8 & 1 \end{pmatrix} = \begin{pmatrix} 61 & 33 \\ -33 & 61 \end{pmatrix}$ representa $z_1 \cdot z_2 = 61 - 33i$.

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

1. Solicitar al usuario que introduzca por consola las partes real e imaginaria de dos números complejos, y leer los valores introducidos.
2. Declarar dos variables de tipo `std::complex`, inicializándolas a los valores introducidos por el usuario. Escribir en la consola el resultado de realizar la suma y el producto de estas dos variables complejas.
3. Declarar dos arrays de dos dimensiones llamados `M` y `N`, almacenando en ellos las matrices 2×2 que representan los dos números complejos introducidos por el usuario.
4. Calcular la suma y el producto de estas dos matrices empleando los dos arrays y escribir en la consola los resultados.
5. Terminar.

Ejercicio 3

Un fichero de texto llamado *diarioActividad.txt* contiene el registro de la actividad de varias máquinas a lo largo de un determinado día. Cada fila del fichero describe una actividad realizada por una máquina. Cada actividad tiene un instante de inicio y un instante de finalización. Cada máquina tiene su propio identificador, el cual es una cadena de caracteres formada por la letra M seguida de un número entero comprendido entre 1 y 99.

El fichero contiene cuatro columnas, cuyo significado se explica a continuación.

- La primera columna contiene el identificador de la máquina que ha realizado la actividad.
- La segunda y tercera columnas contienen los instantes en que la máquina comenzó y terminó la actividad. Los instantes de tiempo están escritos en el formato HH:MM, donde la hora del día (HH) varía entre 05 y 22, y los minutos (MM) entre 00 y 59.
- La cuarta columna contiene una etiqueta que proporciona información adicional sobre la actividad. Los tres posibles valores de esta etiqueta son: AUTO, MANUAL y ERROR.

A continuación se muestra un ejemplo de cuál podría ser el contenido del fichero.

```
M53 13:13  14:02 MANUAL
M64 12:22  14:23 MANUAL
M53 11:45  12:10 MANUAL
M53 07:13  09:14 AUTO
M62 14:40  15:22 ERROR
M64 15:57  18:11 MANUAL
M53 20:05  21:42 AUTO
M64 06:25  06:40 MANUAL
M64 20:38  21:41 MANUAL
M53 16:19  18:54 MANUAL
M64 08:11  09:23 AUTO
M2  13:12  15:01 MANUAL
M53 11:03  11:10 ERROR
M15 11:00  11:30 AUTO
M2  17:47  19:47 AUTO
M2  20:16  21:45 MANUAL
M15 14:00  15:15 AUTO
M64 21:50  21:55 AUTO
M2  06:02  10:11 AUTO
M53 05:23  05:40 AUTO
M53 06:20  06:20 ERROR
M53 06:43  07:01 AUTO
```

Escriba un programa en C++ que lea el fichero de texto *diarioActividad.txt*, y que calcule y muestre en la consola, para cada máquina:

- El tiempo total acumulado, expresado en minutos, durante el cual la máquina ha estado realizando actividades etiquetadas como AUTO.
- El número total de actividades que ha realizado la máquina con cada una de las tres etiquetas (AUTO, MANUAL y ERROR).

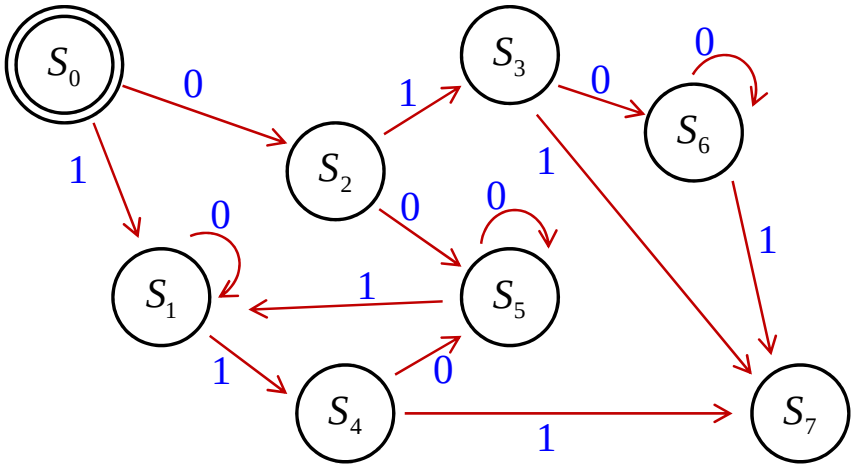
El programa debe escribir la información anterior ordenando las máquinas por orden creciente de su número de identificador. En el caso del fichero ejemplo mostrado anteriormente, este orden sería: M2, M15, M53, M62 y M64.

Al escribir el programa, puede asumir que el fichero *diarioActividad.txt* no contiene errores de formato. Sin embargo, no asuma que las líneas del fichero están escritas siguiendo un determinado orden (por ejemplo, por identificador de máquina, por orden cronológico, etc.), ya que en general estarán desordenadas.

Muestre en la memoria el resultado obtenido al ejecutar su programa empleando un fichero *diarioActividad.txt* cuyo contenido sea el del ejemplo dado anteriormente.

Ejercicio 4

Consideremos un sistema que tiene una entrada, a través de la cual recibe una secuencia de dígitos binarios. El sistema se encuentra inicialmente en el estado S_0. En función de la entrada recibida, el sistema va transitando por los estados $\{S_0, \ldots, S_7\}$ de la manera indicada en el diagrama de estado de la figura mostrada a continuación.



Cada flecha indica una transición en el estado del sistema. En algunos casos, el estado final de la transición es el mismo que el inicial. La etiqueta (de valor 0 o 1) de cada flecha indica el valor que debe tener la entrada para que se produzca la correspondiente transición. Obsérvese que no hay ninguna transición que tenga como estado inicial S_7, por lo cual una vez alcanzado ese estado el sistema permanece en él indefinidamente.

La explicación siguiente pretende clarificar lo anterior.

- El sistema inicialmente está en el estado S_0. Al recibir un dígito de entrada, cambia al estado S_1 o S_2 dependiendo del valor del dígito: cambia al estado S_2 si el dígito de entrada vale 0 y al estado S_1 si vale 1.
- Si estando en el estado S_1 el sistema recibe un dígito de entrada, permanece en el estado S_1 o cambia al estado S_4 dependiendo de si el valor de dicho dígito es 0 o 1 respectivamente.
- ... y así sucesivamente. Una vez el sistema alcanza el estado S_7, permanece en él indefinidamente.

Las entradas al sistema se generan empleando un generador de números pseudoaleatorios distribuidos uniformemente en el intervalo $[0, 1]$. Si el número generado es menor que 0.5, la entrada al sistema es 0 y en caso contrario es 1.

Se puede realizar el experimento siguiente. Partiendo del estado inicial S_0 y generando las entradas de la forma indicada anteriormente, es posible contar el número de transiciones de estado que se producen hasta que el sistema alcanza el estado S_7. Sea T este número de transiciones.

Pueden realizarse N réplicas del experimento anterior, empleando N secuencias independientes de números pseudoaleatorios. De esta forma, se obtienen N valores del número de transiciones.

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

1. Declarar una constante global Booleana llamada `MODO_DEBUG` y asignarle el valor *true*. Como se especifica más adelante, el valor de esta constante condiciona la salida por consola del programa.

2. Declarar una constante global entera `N` y asignarle el valor 10. Esta constante describe el número N de réplicas del experimento que el programa debe ejecutar.

3. Realizar N réplicas del experimento descrito anteriormente, empleando N secuencias independientes de números pseudoaleatorios. El programa debe almacenar en un array unidimensional llamado `T`, de N componentes, el número total de transiciones de estado que se producen en cada una de las N réplicas del experimento.

4. Si la constante `MODO_DEBUG` vale *true*, el programa debe escribir en la consola la secuencia de estados y entradas en cada uno de los experimentos, así como el contenido del array `T`.

5. Sea T_{max} el mayor valor almacenado en el array `T`. Sea p_i el número de veces que aparece en el array `T` en valor i, con $i = 0, \ldots, T_{max}$. Para cada uno de los valores de p_i mayores que cero, con $i = 0, \ldots, T_{max}$, escribir en la consola una línea de texto con el formato siguiente:

   $< i >$ `transiciones en` $< p_i >$ `replicas`

   donde $< p_i >$ y $< i >$ representan los correspondientes valores numéricos de p_i e i respectivamente.

6. Terminar.

A continuación se muestra un ejemplo de ejecución del programa para los valores siguientes de las constantes: `MODO_DEBUG = true`, `N`= 10. Dado que las entradas al sistema se generan aleatoriamente, una nueva ejecución del programa proporcionaría resultados diferentes.

```
Evolucion del automata en 10 replicas
S0, 1, S1, 1, S4, 1, S7
S0, 0, S2, 0, S5, 1, S1, 1, S4, 0, S5, 1, S1, 0, S1, 0, S1, 0, S1, 1, S4, 1, S7
S0, 0, S2, 0, S5, 0, S5, 1, S1, 0, S1, 1, S4, 1, S7
S0, 1, S1, 0, S1, 1, S4, 1, S7
S0, 0, S2, 0, S5, 1, S1, 0, S1, 1, S4, 1, S7
S0, 1, S1, 0, S1, 0, S1, 0, S1, 0, S1, 1, S4, 0, S5, 1, S1, 0, S1, 1, S4, 1, S7
S0, 1, S1, 1, S4, 0, S5, 0, S5, 0, S5, 0, S5, 0, S5, 1, S1, 0, S1, 1, S4, 1, S7
S0, 1, S1, 0, S1, 1, S4, 0, S5, 0, S5, 1, S1, 1, S4, 0, S5, 0, S5, 0, S5, 0, S5, 1, S1, 0, S1, 1, S4, 1, S7
S0, 1, S1, 1, S4, 1, S7
S0, 0, S2, 1, S3, 0, S6, 1, S7

Contenido del vector T:
3 11 7 4 6 11 11 15 3 4

Informe sobre el numero de transiciones:
3 transiciones en 2 replicas
4 transiciones en 2 replicas
6 transiciones en 1 replicas
7 transiciones en 1 replicas
11 transiciones en 3 replicas
15 transiciones en 1 replicas
```

A continuación se muestra otro ejemplo de ejecución del programa, en este caso asignando los siguientes valores a las constantes: `MODO_DEBUG = false`, `N`= 10000.

```
Informe sobre el numero de transiciones:
3 transiciones en 2396 replicas
4 transiciones en 1262 replicas
5 transiciones en 893 replicas
6 transiciones en 801 replicas
7 transiciones en 650 replicas
8 transiciones en 529 replicas
9 transiciones en 469 replicas
10 transiciones en 359 replicas
11 transiciones en 330 replicas
12 transiciones en 282 replicas
13 transiciones en 262 replicas
14 transiciones en 212 replicas
15 transiciones en 180 replicas
16 transiciones en 172 replicas
17 transiciones en 149 replicas
18 transiciones en 121 replicas
19 transiciones en 131 replicas
20 transiciones en 91 replicas
21 transiciones en 82 replicas
22 transiciones en 92 replicas
23 transiciones en 68 replicas
24 transiciones en 59 replicas
```

```
25 transiciones en 48 replicas
26 transiciones en 48 replicas
27 transiciones en 45 replicas
28 transiciones en 26 replicas
29 transiciones en 26 replicas
30 transiciones en 22 replicas
31 transiciones en 27 replicas
32 transiciones en 20 replicas
33 transiciones en 16 replicas
34 transiciones en 16 replicas
35 transiciones en 11 replicas
36 transiciones en 13 replicas
37 transiciones en 9 replicas
38 transiciones en 11 replicas
39 transiciones en 7 replicas
40 transiciones en 9 replicas
41 transiciones en 6 replicas
42 transiciones en 7 replicas
43 transiciones en 7 replicas
44 transiciones en 8 replicas
45 transiciones en 6 replicas
46 transiciones en 3 replicas
47 transiciones en 4 replicas
48 transiciones en 2 replicas
49 transiciones en 2 replicas
51 transiciones en 2 replicas
52 transiciones en 1 replicas
54 transiciones en 1 replicas
55 transiciones en 1 replicas
56 transiciones en 2 replicas
61 transiciones en 1 replicas
62 transiciones en 1 replicas
63 transiciones en 1 replicas
79 transiciones en 1 replicas
```

Muestre en la memoria el resultado de ejecutar dos veces el programa que usted ha realizado, asignando el siguiente valor a las constantes:

- **Prueba 1:** `MODO_DEBUG = true`, `N` $= 10$
- **Prueba 2:** `MODO_DEBUG = false`, `N` $= 10000$