UNIVERSIDAD NEBRIJA

PROGRAMACIÓN II - GRUPO C

PRÁCTICA Nº3

Miércoles, 23 de marzo 2023

INSTRUCCIONES

1. Sólo se podrá utilizar una hoja con el “chuletario” elaborado por cada alumno con sintaxis del lenguaje. No se permitirá la consulta de ningún otro tipo de material durante la realización de la práctica.

2. La entrega de la práctica sólo se admitirá a través de la **actividad** disponible en el campus virtual de la asignatura de prácticas de PII del grupo C **antes de la hora de finalización de la sesión de prácticas**.

3. Aunque las prácticas se tengan que realizar en grupos de dos integrantes, para su evaluación, **ambos deberán hacer la entrega a través de su campus virtual, la misma práctica**. En otro caso, la práctica quedará **sin evaluar** y supondrá un **0 en la calificación del estudiante que no la haya entregado**.

4. Se debe entregar **los ficheros** correspondientes al ejercicio en formato **.cpp, sin comprimir**.

5. Los ficheros entregados deben **incluir, al principio del cada fichero entregado, el nombre de los integrantes del equipo**.

6. **El incumplimiento de alguna de las instrucciones sobre la realización/entrega de la Práctica supondrá su descalificación.**

Criterios generales de evaluación

Funciones/Métodos: Si no se usa el paso por referencia constante cuando las variables de los parámetros de entrada no son de tipo simple.	40%
Tipos de datos y variables: • Uso de variables globales (fuera del ámbito de una función). • Si no se usan los tipos contenedor vistos en clase (std::array; std::vector; std::set; std::string, etc.) para las variables que lo necesiten.	 0% 0%
El **programa no compila** o **no se asemeja** a lo pedido**.**	0%
Si no se cumplen los **criterios de entrega** indicados en la **actividad/examen.**	0%

Criterios particulares de evaluación

El elemento evaluable no compila o no se asemeja a lo que se pide	0%
El elemento evaluable no se aproxima suficientemente a lo pedido	40%
El elemento evaluable se aproxima suficientemente a lo pedido	60%
El elemento evaluable funciona correctamente y las estrategias y elementos de código elegidos son adecuados.	100%

IMPORTANTE:

- Todos los ejercicios del examen deberán ser resueltos de forma **algorítmica**, es decir, la **solución** propuesta tendrá que ser **general** y **no particular** para unos determinados datos/valores.

- Todos los **ejercicios resueltos sin utilizar funciones** cuando sea apropiado se valorarán con una **nota máxima del 60% sobre la calificación prevista**.

- Se recomienda una **primera lectura del enunciado de la práctica** para planificar su realización.

- Conviene ir **probando los diferentes recursos programados** de forma gradual. Programar todo seguido y realizar las pruebas al final cuando quedan 10 minutos para la entrega, suele acabar con **errores de compilación sin resolver en la entrega** y por lo tanto con una calificación de 0 puntos.

Práctica *PII_P3GC*

Esta práctica consiste en desarrollar en C++11 los elementos adecuados para que el código correspondiente al **programa** y **ficheros cabecera proporcionados** en la actividad funcionen correctamente y cumplan con las especificaciones que se indican.

IMPORTANTE: Se pueden comentar las partes del programa que no se consigan implementar para **evitar errores de compilación**, en cuyo caso, la evaluación de la práctica tendrá su correspondiente **penalización proporcional a la parte que no se haya desarrollado**.

Complejo: (3 puntos)

Crear esta clase implementando los métodos de los que se proporciona la interfaz en el fichero ***claseComplejo.h*** y y que se describen a continuación:

- **Constructor por defecto** que inicializará los atributos a cero. Implementar con inicializadores.
- **Constructor con parámetros** para dar valor a los atributos. Implementar con inicializadores.
- **Métodos gets** para cada atributo.
- **Método restarValor** que decrementará el valor de los atributos del objeto del número complejo que lo invoca, en la cantidad recibida a través del parámetro. Ejemplo, si nC1=5.7+10,6i y el parámetro de restarValor es 2, el resultado será que nC1=3.7+8,6i.
- **Operador decremento (--)**, tanto en la forma prefija como postfija, ambos sobrecargados. El **operador decremento en forma prefija** decrementará en una unidad ambos atributos del número complejo. Ejemplo:

  si nC1=5.7+10,6i entonces, su resultado será: si nC1=4.7+9,6i

  El **operador decremento en forma postfija**, además de decrementar en una unidad ambos atributos del objeto que lo invoca, permitirá que se asigne el valor a otro objeto de la clase Complejo antes de decrementar el objeto implícito. Ejemplo:

  si nC1=5.7+10,6i y la operación es: nC2 = nC1—

  el resultado será: nC2 = 5.7+10,6i y nC1=4.7+9,6i

Además, como **funciones generales** se tendrán que implementar el **operador resta (-)** y el **operador <<**, ambos sobrecargados. El operador resta (-) permitirá restar dos números complejos recibidos como parámetros y devolverá el resultado sin modificarlos. El operador << servirá para mostrar por pantalla los componentes del número complejo.

Matriz: (5 puntos)

Crear esta clase implementando los métodos de los que se proporciona la interfaz en el fichero ***claseMatriz.h*** y que se describen a continuación:

- **Constructor con los parámetros** filas y columnas para dar valor a los respectivos atributos que dimensionan la matriz y el atributo de tipo vector. Si alguna de las dos dimensiones no son valores válidos, el método debe lanzar **una excepción de tipo int con valor 1** para luego mostrar por pantalla el mensaje: "**ERROR, las dimensiones son incorrectas**".
- **Métodos gets** únicamente para los atributos **filas** y **columnas**.
- **Método pos** que recibirá como parámetro el valor de la fila y columna de un elemento de la matriz y tendrá que devolver el valor del elemento almacenado en el vector. Hay que tener en cuenta que el vector atributo de la clase es de una dimensión, por tanto habrá que calcular cuál sería la posición correspondiente en base a la fila y columna que se indican. Si alguno de los dos parámetros, fila o columna, no son valores válidos respecto a las dimensiones establecidas al crear el objeto matriz, el método debe lanzar **una excepción de tipo int con valor 2** para luego mostrar por pantalla el mensaje: "**ERROR, la posicion del elemento no es correcta**".
- **Método restarValor** que decrementarán, en la cantidad recibida a través del parámetro, los valores guardados en el atributo de tipo vector. Esta operación es semejante a la definida en la clase Complejo.
- **Operador decremento (--)**, tanto en la forma prefija como postfija, ambos sobrecargados. El **operador decremento en forma prefija** decrementará en una unidad los valores guardados en el atributo de tipo vector. El **operador decremento en forma postfija**, además de decrementar en una unidad los valores guardados en el atributo de tipo vector del objeto que lo invoca, permitirá que se asigne el vector a otro objeto de la clase, antes de

decrementar los datos del vector del objeto implícito. Estos operadores son semejantes a los definidos en la clase Complejo.

Además, hay que implementar el **operador resta (-)** como **función general**. Este operador sobrecargado permitirá restar dos matrices recibidas como parámetro y devolverá el resultado de dicha operación sin modificar las matrices dato. Si las dimensiones de las matrices no son compatibles para realizar la resta, la función debe lanzar **una excepción de tipo int con valor 3** para luego mostrar por pantalla el mensaje: "**ERROR, las dimensiones de las matrices no son correctas**".

Finalmente, como **funciones generales**, sobrecargar el **operador <<** para mostrar por pantalla la matriz con el formato de filas y columnas, y el **operador >>** para leer del teclado, por filas, los valores de la matriz que indique el usuario.

Por último, **templatizar** las funciones generales, **funcion1** y **funcion2,** que se tendrán que incluir en el fichero ***Funciones.h,*** para que reciban **objetos** de las **clases Complejo y Matriz**, devolviendo el resultado adecuado según corresponda. **(2 puntos)**

- **funcion1:** deberá permitir restar dos objetos recibidos como parámetros.
- **funcion2:** deberá permitir decrementar los componentes del objeto que recibirá como parámetro en una cantidad que también se recibirá como parámetro. **NOTA:** cuando el objeto sea una matriz, solo se deben decrementar los valores almacenados en el atributo de tipo vector.