UNIVERSIDAD NEBRIJA

PROGRAMACIÓN II - GRUPO C

PRÁCTICA Nº2

Miércoles, 2 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 A **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**, con el nombre de ***PII_P2GC***.

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_P1GC*

Esta práctica consiste en desarrollar en C++11 un programa que permita trabajar de forma sencilla con **vectores numéricos (1 dimensión)**. Para ello se realizará la **sobrecarga de operadores** y el **manejo de excepciones** respondiendo a las siguientes especificaciones.

Se presentará un menú que permita al usuario seleccionar la operación que desea realizar. Según la opción seleccionada, se **solicitará al usuario tantos vectores numéricos como sean necesarios para realizar dicha operación**. El resultado de cada operación se mostrará por pantalla al usuario. El programa terminará cuando la opción seleccionada del menú sea **FIN**.

1. Restar dos vectores componente a componente
2. Decrementar cada componente del vector.
3. Comparar dos vectores por desigualdad.
0. FIN

El programa principal (***main***) tendrá que declarar varios objetos de la **clase CVector** que se describe más adelante. Las operaciones que se podrán realizar con dichos objetos son las siguientes:

La **opción 1**, utilizará **dos objetos** de la **clase CVector**. Solicitará los datos necesarios al usuario para que ambos objetos tengan valores. Después, utilizando el **operador - sobrecargado** y **siempre que sea posible**, se realizará la resta de ambos vectores **dejando el resultado en un nuevo vector**. Para finalizar esta opción, **desde el programa principal (main)**, se mostrará por pantalla el contenido de los atributos de **todos los vectores** implicados (los dos operandos y el resultado de la resta). (**0,5 puntos**)

La **opción 2,** utilizarán **dos objetos** de la **clase CVector**, V1 y V2. Solicitará los datos necesarios al usuario para que uno de dichos objetos tenga valores. Después, utilizando el **operador -- sobrecargado**, se realizará la siguiente operación:

V1=V2--;

Que tendrá como resultado que V1 tenga los mismo valores que V2 antes del decremento y que V2 **decremente cada uno de sus componentes**. Para finalizar esta opción, **desde el programa principal (main)**, se mostrará por pantalla el

contenido de los atributos de los **vectores implicados en la operación anterior**. (**0,5 puntos**)

La **opción 3**, utilizará **dos objetos** de la **clase CVector**. Solicitará los datos necesarios al usuario para que ambos objetos tomen valor. Después, utilizando el **operador != sobrecargado** y **siempre que sea posible**, se realizará la **comparación dichos objetos** (componente a componente de cada uno de los vectores) para determinar si el contenido de los dos vectores son distintos o no, indicando el resultado a través de un valor booleano. Para finalizar esta opción, **desde el programa principal (main)**, se mostrarán por pantalla el contenido de los atributos de **ambos vectores** y un **menaje con el mensaje adecuado al resultado de la comparación**. (**0,5 puntos**)

Para organizar la información y poder reutilizarla en futuras aplicaciones se tendrá que definir la siguiente clase:

Clase ***CVector***

- Atributos:
  - Una variable que indique el tamaño del vector numérico, es decir, el **número de valores** que almacena.
  - Una variable, del tipo que se considere adecuado, para almacenar los valores numéricos (tipo ***float***) del vector.
- Métodos:
  - **Constructor por defecto**. **(0,25 puntos)**
  - **Constructor que tenga como parámetro el tamaño del vector**. Este parámetro debe servir para dar valor y dimensionar (dar tamaño) a las variables encapsuladas en la clase, respectivamente. **(0,5 puntos)**
  - **Constructor de copia**. **(0,5 puntos)**
  - Un método **getSize** que devuelva el atributo correspondiente al **número de valores** guardados en el vector. **(0,25 puntos)**
  - Un método **getNum** que a partir de una posición correcta del vector devuelva el **valor del número almacenado en dicha posición**. En caso de intentar acceder a una posición que no exista en el vector, el método debe lanzar **una excepción de tipo int con valor 1**. (**0,75 puntos**)
  - Un método **setNum** que a partir de dos argumentos (posición y valor numérico) se asigne el valor indicado en la posición dada del vector. En el

caso de intentar acceder a una posición que no exista en el vector, el método debe lanzar **una excepción de tipo int con el valor 2**. (**0,75 puntos**)

- Un método **setSize** que a partir de un argumento (dimensión para el vector) se asigne el valor indicado al número de valores del vector y se dimensione el vector para poder guardar esos datos. (**0,5 puntos**)
- **Operadores** que se deben de sobrecargar dentro de la clase **CVector** para facilitar y realizar ciertas operaciones sobre los vectores numéricos:
  - ✓ **Operador -**, este operador debe permitir realizar la **resta de dos objetos de la clase CVector** devolviendo el resultado de la operación sin cambiar el estado de los objetos utilizados como operandos (vectores restados). Si los vectores no se pueden restar, se debe lanzar **una excepción de tipo int con el valor 3**. (**1,5 puntos**)
  - ✓ **Operador --** (decremento postfijo), el operador debe trabajar como un operador postfijo de C++, permitiendo asignar el contenido del objeto antes de proceder a **decrementar en una unidad cada componente del vector**. (**1,5 puntos**)

Además, como **función general**, fuera de la clase CVector, se tiene que definir e implementar el **operador !=** (compara por desigualdad). Este operador debe determinar **si dos objetos de la clase CVector contienen o no distintos valores**, devolviendo un valor bool. Si los vectores no se pueden comparar, se debe lanzar **una excepción de tipo int con el valor 4**. (**1,5 puntos**)

Finalmente, se debe diseñar e implementar una **función general** llamada ***mostrar*** que, a partir de un **puntero inteligente compartido a un objeto CVector**, permita **visualizar por pantalla los elementos del vector y su tamaño**. (**0,5 puntos**)