UNL FICH Facultad de Ingeniería y Ciencias Hídricas

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIDAD 02

Introducción a la Programación Orientada a Objetos

Guía de trabajos prácticos
2023

UNIDAD 02

Introducción a la Programación Orientada a Objetos

Ejercicio 1

Diseñe una clase **Cilindro** que modele un cilindro con el objetivo de calcular el volumen del cuerpo conociendo el radio de su base y la altura del mismo.

a. Cree los métodos *AsignarDatos(...)* y *ObtenerVolumen(...)* para asignar los datos del problema y obtener el volumen del cuerpo.
b. Escriba un programa cliente que utilice dicha clase. Defina 2 instancias de **Cilindro** llamadas *c1* y *c2*. El objeto *c1* debe utilizar datos ingresados por el usuario, mientras que para *c2* utilice 5.3 cm y 10.2 cm para el radio y la altura respectivamente.
c. Agregue un constructor a la clase **Cilindro** que reciba 2 parámetros para inicializar el radio y la altura. Luego intente compilar nuevamente el programa. ¿Puede explicar por qué se produce un error? Proponga una solución.

Ejercicio 2

Proponga una clase **EcuacionCuadratica** para modelar ecuaciones cuadráticas de la forma $ax^2+bx+c=0$. La clase debe incluir:

a. Un constructor que reciba los valores de los coeficientes *a, b y c*.
b. Un método *TieneRaicesReales(...)* que devuelva verdadero si las raíces de la ecuación son reales.
c. Dos métodos *VerRaiz1(...)* y *VerRaiz2(...)* que permitan obtener las raíces reales (en caso de que lo sean).
d. Dos métodos *VerParteReal(...)* y *VerParteImag(...)* que permitan obtener las partes real e imaginaria de las raíces complejas (en caso de que lo sean).
e. Cree un programa cliente que utilice un objeto de la clase **EcuaciónCuadratica** para determinar las raíces de una ecuación cuadrática cuyos coeficientes sean ingresados por el usuario, y las muestre en el formato que corresponda (según sean reales o complejas).

Ejercicio 3

Diseñe una clase **Polinomio** que contenga:

a. un constructor que reciba el grado el polinomio e inicialice sus coeficientes en 0.
b. un método que permita cambiar un coeficiente.
c. un método evaluar que permita evaluar el polinomio para un valor dado de

la variable *x*.

d. el/los métodos que considere necesarios para poder mostrar un polinomio desde un programa cliente.
e. una función que permita sumar dos polinomios retornando un tercer polinomio con el resultado (ej. p_res=Sumar(p1,p2);). ¿Cómo variaría si en lugar de ser una función libre fuera un método de la clase?

Verifique el funcionamiento de la función *Sumar(...)* mediante un programa cliente.

Ejercicio 4

Implemente una clase **VectorDinamico** que posea como atributo un puntero a entero que apunte a la memoria donde se almacenan los datos. Dicha clase debe poseer:

a. Un constructor que reciba el tamaño inicial del vector, reserve la memoria necesaria e inicialice los valores del vector de manera aleatoria.
b. Un destructor que se encargue de liberar la memoria reservada.
c. Un método *Duplicar(...)* que duplique la cantidad de memoria reservada manteniendo los datos que ya estaban en el vector e inicializando al azar los nuevos valores.
d. Un método *VerElemento(...)* que reciba el número de elemento y devuelva su valor.
e. Cree un programa cliente que muestre la utilización de todas las funciones implementadas.
f. Explique: ¿Es necesario implementar un constructor de copia?¿Por qué?

Ejercicio 5

Se dispone del siguiente tipo de dato:

```
struct Alumno {
      string nombre;
      float nota;
};
```

En base al mismo se desea crear una clase **Curso** para modelar el cursado de una materia. La clase deberá contener el nombre de la materia y la cantidad de alumnos en el curso junto con una lista de los mismos. Proponga los siguientes métodos:

a. Constructores y destructor según lo considere conveniente.
b. Un método que permita agregar un Alumno.
c. Un método que determine el promedio del curso.
d. Un método que devuelva la calificación más alta y el nombre del alumno que la obtuvo.

Cuestionario

1. ¿Qué es un objeto? ¿Qué es una clase? ¿Qué es una instancia?
2. ¿Qué es un constructor?
3. ¿Cuándo se invoca al constructor de un objeto?¿Cuándo al destructor?
4. ¿Qué significa constructor por defecto?¿Y constructor de copia?
5. ¿Qué significa encapsulamiento? ¿Qué significa ocultamiento?
6. ¿Cómo se consigue el ocultamiento de datos en C++?
7. ¿Cuál es el efecto de las etiquetas private y public?
8. ¿Cuál es la diferencia, en C++, entre una clase y una estructura? ¿Cuándo se utiliza cada una?
9. ¿Para qué sirve el puntero this?
10. ¿Es posible usar el mismo nombre de función para una función miembro de una clase y una función externa? Si es posible, indique cómo distinguirlas. En caso contrario, indique las razones.
11. ¿Qué implica que un atributo se declare con el modificador static?¿Qué ocurre cuando un método es declarado con dicho modificador?

Ejercicios Adicionales

Ejercicio 1

Cree una clase **Rectangulo** que posea:

a. Un constructor que permita crear el objeto a partir de la base y altura del rectángulo.
b. Otro constructor que permita crear el objeto a partir de las coordenadas (x, y) de dos vértices opuestos.
c. Métodos *VerArea(...)* y *VerPerimetro(...)* que calculen respectivamente el área y perímetro del rectángulo.
d. Un método *EsCuadrado(...)* que permita conocer si el rectángulo es cuadrado.
e. Cree un programa cliente donde utilice varias instancias de la clase **Rectangulo**. Cree los objetos dinámicamente y no olvide borrarlos al terminar de utilizarlos.

Ejercicio 2

Escriba una clase llamada **Fecha** con atributos para definir una fecha (día, mes, año). Implemente los siguientes métodos:

a. Un constructor que reciba el día, mes y año y los inicialice.
b. Métodos para devolver el día, mes y año de la fecha.
c. Una función llamada *DiferenciaEnAnios(...)* que reciba otro objeto de tipo **Fecha** y devuelva la diferencia en años entre ambas.
d. Cree una función llamada *SignoZodiacal()* que devuelva una cadena de texto con el nombre del signo del zodiaco de una persona nacida en esa fecha.

Implemente un programa cliente que utilice dicha clase.

Ejercicio 3

Consulte la bibliografía y averigüe el significado de la palabra reservada *static* y su utilización en atributos de clases. Proponga una clase **CuentaObjetos** que permita llevar cuenta de los objetos que son creados. Para ello defina un constructor y un destructor que notifiquen por pantalla cuando un objeto fue creado/destruido y además muestre la cantidad de objetos que aún se encuentran en memoria. Utilice un programa cliente para probar la clase y realice la creación/destrucción de los objetos de forma dinámica.

Ejercicio 4

1. Intente responder a la siguiente pregunta antes de continuar leyendo este enunciado: ¿Tiene sentido colocar un constructor en la parte privada de una clase?

2. Analice el siguiente código y determine qué particularidad tendrá esta clase:

```
class Singleton {
      Singleton() { /*unico constructor*/ };
      static Singleton *instancia;
public:
      static Singleton *ObtenerInstancia() {
            if(!instancia) instancia=new Singleton();
            return instancia;
      };
      // ...interfaz de la clase...
};
Singleton *Singleton::instancia=NULL;
```

Ejercicio 5

Añada un *move*-constructor a la clase *VectorDinámico*. Declare una instancia *v1* con datos generados aleatoriamente y luego analice las diferencias en la ejecución de "*VectorDinamico v2 = v1;*" y "*VectorDinamico v2 = move(v1);*". Ejecute ambos casos paso a paso mediante un depurador para verificar si su análisis es correcto.