Unidad Nº1: Introducción a la POO.

Preguntas orientadoras

1) ¿Qué entiende por paradigma? ¿Qué paradigma de programación utilizó hasta ahora? ¿Cuáles son sus características?.

2) Describa el tipo de Dato Abstracto (TAD). ¿Puede el lenguaje C trabajar con TAD's?. Dé un ejemplo de tipo de dato abstracto en C si el lenguaje lo soporta o de lo contrario indique como lo haría.

3) ¿En qué se basa la programación orientada a objetos POO? ¿Qué ventajas presenta respecto de otros lenguajes? ¿Qué características salientes posee?.

4) Defina clase y objeto. Explicite de qué manera se puede fijar el nivel de acceso a los miembros de una clase.

5) ¿Cómo se definen las funciones o métodos en una clase?.

6) Defina Lenguaje Unificado de Modelado. ¿Qué tipos de diagramas UML conoce?. Cree una clase, defina la visibilidad de los atributos y métodos, genere 3 clases adicionales que hereden métodos y atributos de la clase anterior. Genere para esa situación un Diagrama UML.

7) En C, para el manejo de archivos, utilizó *File** el cual es un *TDA, investigue al respecto.*

Ejercicios

1) Genere un proyecto de Consola de C++ y comente las diferencias y similitudes que nota con un proyecto de consola de C.

2) Escriba una función en C++ que permita intercambiar los valores de dos variables enteras pasadas por referencias desde el main. Este problema ya se resolvió en programación estructurada usando punteros. ¿Cuál es la diferencia entre un puntero y una referencia? ¿Le resultó más fácil la implementación usando las referencias?.

3) El siguiente fragmento de código muestra cómo almacenar el resultado de la división entre dos números enteros. Escriba dicha conversión en lenguaje C++:

```
int a,b; float r; a=5; b=2; r = (float) a /b;
```

4) Realice un programa que permita saber las veces que una función fue invocada desde el programa principal.

5) Implementar un TDA llamado Vehículo, organizando la definición en un archivo de cabecera y su implementación en un archivo .c. Utilice una estructura que contenga campos para representar los atributos del Vehículo: marca, puertas, kilometraje y cilindrada y punteros a funciones: "*getters*" y "*setters*" para acceder a los atributos antes mencionados y algunas acciones que puede realizar el Vehículo: acelerar, frenar, prender, apagar. Además, son necesarias dos funciones encargadas de reservar y liberar memoria de forma dinámica: crearVehículo y destruirVehículo.

6) Implementar el mismo TDA del ejercicio 5, pero ahora usando una clase (*class*) y una estructura (*struct*) de C++. ¿Qué diferencias y qué similitudes nota entre una "*class* y una *struct* de C++? ¿Y entre una struct de C y una de C++? ¿Experimente con los modificadores de acceso sobre métodos y atributos? ¿Le resultó más fácil implementar el TDA en C++? ¿Por qué?

7) Agregue un atributo público estático llamado "valor_patente" de tipo float a la Clase Vehículo del ejercicio 6. Luego, desde el programa principal, cree dos representantes o "instancias" de tipo Vehículo y en una de las instancias cambie el valor de dicho atributo. Entonces, si usando un "*printf*" muestra el valor del mismo atributo, pero de la otra instancia, ¿Qué valor muestra se muestra en consola? ¿Por qué? ¿Será posible acceder al atributo en cuestión SIN crear ninguna instancia de Vehículo? Experimente y comente.

8) Antes de que el genio de Stroustrup inventara el lenguaje C++ la gente escribió mucho código en C. ¿Conoce la forma de usar código escrito en C desde C++? En la clase teórica se mencionó la palabra reservada "*extern*". Entonces, escriba un conjunto de funciones en C, (los prototipos en un archivo .h y la implementación en .c), para sumar, restar, multiplicar y dividir enteros. Luego, escriba un programa en C++ que use dichas funciones.