**Programación Orientada a Objetos Parcial 1 - Tema A 29/09/2015**

Nota: Falta el ejercicio 1, porque correspondía a un tema que este (2017) año no se dió.

2) Programar una clase **Animal** que tenga un método *desplazarse()* y otro método *tipo()*. Programe las clases **Mamifero**, **Ave** y **Pez**, que son hijas de la clase **Animal**. Los atributos de las clases son el tipo y la especie del animal, cuyos valores deben ser pasados al constructor de la clase base. Además deberá sobreescribir los métodos *desplazarse()* y *tipo()*, de tal manera que el primer método devuelve una cadena de texto indicando el tipo de desplazamiento según el tipo: “camina” para el tipo “mamífero”, “vuela” para el tipo “ave” y “nada” para el tipo “pez”. Escriba un programa cliente que permita modelar un **Parque** y almacene la información de los animales en un único arreglo de 30 elementos y luego emita un listado de los mismos describiendo el tipo y la forma de desplazamiento de cada animal. Finalmente responda: ¿Hay entre las clases algún método virtual puro?¿Por qué?

3) Escriba una clase llamada **Fecha** con los atributos necesarios para definir una fecha (día, mes, año). Proponga, además de los métodos que considere necesarios, una sobrecarga del operador <= que permite comparar 2 fechas y una sobrecarga del operador [] para obtener el día, mes y año con los valores 1, 2 y 3 respectivamente. Además, sobrecargue el operador >> para poder leer una fecha desde la consola. Finalmente cree un programa cliente que utilice las sobrecargas implementadas.

4) a)Cuál es el objetivo de **this**? b) Qué es un método virtual puro?; c) Qué es encapsulamiento en POO? d) Es necesario sobrecargar el operador = para una clase? e) complete el código en los puntos suspensivos : int *p; …… ; cout<<*(p+i); de modo de definir dinámicamente un arreglo de 20 enteros al azar menores que 500 y mostrarlos

**Programación Orientada a Objetos Parcial 1 - Tema B 29/09/2015**

Nota: Falta el ejercicio 1, porque correspondía a un tema que este (2017) año no se dió.

2) Escriba una clase llamada CORREOS con los siguientes atributos: Nombre y mail (ambos string). Proponga entre los métodos las siguientes sobrecargas: a) operador == para comparar los objetos de tipo CORREOS por sus nombres, b) operador [] para que devuelva el nombre o el email de acuerdo al índice 1 o 2 respectivamente, y c) operador << para poder mostrar y mail de una instancia x mediante cout<<x.

3) Programar una clase Alumno y otra Docente, que son hijas de una clase Persona, que tiene el método Saludar. Los atributos de un alumno son el nombre y la carrera a la que está inscripto. Los atributos del docente son el nombre y la materia que dicta. En las clases hijas se debe sobreescribir el método Saludar, de tal manera que si cuando un docente saluda, debe devolver “Mi nombre es XXXXXX y dicto WWWWWW” (donde XXXXXX es el nombre y WWWWW la materia), mientras que si saluda un alumno, deberá devolver “soy ZZZZZZ y pertenezco a la carrera TTTTTT”. Escriba un programa cliente que tenga en un único vector un docente y 24 alumnos, y un método para que todos saluden. Finalmente responda: ¿Hay en esta jerarquía de clases alguna clase abstracta?¿Por qué?

4) Explique: a) ¿Qué significa que una clase A es contenedora de otra B? b) Antes de codificar en C++, ¿como reconoce si 2 clases pueden relacionarse a través de herencia¿ c) ¿Qué entiende por ocultación y cómo se implementa en C++? d) ¿Cuando es necesario sobrecargar el constructor de copia de una clase?

**FICH-UNL 1ER PARCIAL POO 4/10/2016 - TEMA -A-**

Nota: Falta el ejercicio 1, porque correspondía a un tema que este (2017) año no se dió.

**Ejercicio 2 (25 pts)**

a) Diseñe una clase para representar un **cilindro** con sus atributos radio y altura, un constructor que inicialice ambos, y un método para obtener su volúmen. Agregue sobrecargas de: b) el operador **[ ]** para acceder a radio (cuando recibe 0) y altura (cuando recibe 1), c) el operador **>** para comparar dos cilindro por volumen (un cilindro será menor que otro si su área es menor); d) el operador **>>** para que "cin>>c" lea el radio y la altura del cilindro c.

**Ejercicio 3 (30 pts)**

Una fábrica de conservas hace envases de forma cilíndrica (lata) o de prisma recto (caja); en ambos envases debemos rotular el volumen en cm3 y el peso en gramos.

Modele una clase base Envase con los atributos volumen y peso. Un método público AsignarPeso(p), un método virtual puro CalcularVolumen() que calcule el volumen de acuerdo a los parámetros de los hijos y otros 2 métodos para VerVolumen() y VerPeso().

Modele la clase hija Lata que tendrá los atributos radio y altura, cuya fórmula de volumen es: área de la base x altura, donde el área de la base es PI*radio^2; y otra clase hija Caja que tendrá los atributos largo, ancho y alto, cuya fórmula de volumen es: largo x ancho x alto. Los atributos los debe cargar en sus respectivos constructores. Estos (los constructores de Caja y Lata) recibirán las medidas necesarias, y el peso.

En el programa principal debe usar un único puntero de tipo Envase para crear primero una Lata y mostrar su volúmen, y luego una Caja y también mostrar su volúmen.

**Ejercicio 4 (20 pts)**

a-¿Qué es un objeto? ¿Qué es una clase? b-¿Cuándo se invoca al constructor de un objeto? ¿Cuándo al destructor? c-¿Cuál es la diferencia, en C++, entre una clase y un struct? ¿Cuándo se utiliza cada una? d-¿Qué implica que un atributo se declare con el modificador static? Ejemplifique.

**FICH-UNL 1ER PARCIAL POO 4/10/2016 - TEMA -B-**

Nota: Falta el ejercicio 1, porque correspondía a un tema que este (2017) año no se dió.

**Ejercicio 2 (25 pts)**

a) Diseñe una clase para representar un **rectángulo** con sus atributos base y altura, un constructor que inicialice ambos, y un método para obtener su área. Agregue sobrecargas de: b) el operador **[ ]** para acceder a base (cuando recibe 0) y altura (cuando recibe 1), c) el operador **<** para comparar dos rectángulos por área (un rectángulo será menor que otro si su área es menor); c) el operador ***** para multiplicar un rectángulo por un escalar (float) y obtener un nuevo rectángulo cuya base y altura sean las del rectángulo original multiplicadas por dicho escalar.

**Ejercicio 3 (30 pts)**

Una fábrica de Tanques los hace de forma de **Cilindro** o de **Esfera**, en ambos envases debemos rotular el volumen en m3 y el peso en kilogramos.

Modele una clase base **Tanque** con los atributos volumen y peso. Un método público AsignarPeso(p), un método virtual puro CalcularVolumen() que calcule el volumen de acuerdo a los parámetros de los hijos y otros 2 métodos para VerVolumen() y VerPeso().

Modele la clase hija **Cilindro** que tendrá los atributos radio y altura, cuya fórmula de volumen es: área de la base x altura, donde el area de la base se calcula como PI*radio^2; y otra clase hija **Esfera** que tendrá el atributo radio, cuya fórmula de volumen es: 4/3*PI*radio^3. Los atributos (medidas y peso) los debe cargar con un constructor.

En el programa principal debe usar un único puntero de tipo **Tanque** para crear primero un **Cilindro** y mostrar su volúmen, y luego una **Esfera** y también mostrar su volúmen.

**Ejercicio 4 (20 pts)**

a) ¿Qué es herencia múltiple? b) ¿Para qué sirve la palabra reservada virtual? c) ¿Qué es una clase abstracta? d) ¿En qué se diferencian agregación y herencia? e) Indique dos formas de obtener una dirección de memoria válida para un puntero.

**Programación Orientada a Objetos - Recuperatorio 1er Parcial - 18/11/2016 - FICH-UNL**

**Ejercicio 1 (25 pts)** Escriba una clase *Vector3D* para representar un vector de 3 componentes *{x;y;z}*. Implemente sobrecargas para los operadores ***[ ]***, **+**, **==** y *****. El operador ***[ ]*** debe permitir además modificar las componentes. Nota: dados dos vectores *{x1;y1;z1}* y *{x2;y2;z2}*, la suma es un vector *{x1+x2 ; y1+y2 ; z1+z2}*, y el producto, un escalar *x1*x2+y1*y2+z1*z2*.

**Ejercicio 2 (25 pts)** Escriba una función que reciba un puntero indicando el comienzo de un arreglo de enteros, un entero indicando la cantidad de elementos en el arreglo, y un tercer parámetro con un valor entero adicional x. La función debe buscar dicho elemento (x) dentro del arreglo y retornar un puntero al mismo (si aparece varias veces, a una cualquiera de ellas). Si el valor no está en el arreglo debe retornar el puntero nulo. Escriba un programa cliente para probar la función que, en caso de encontrar al elemento informe además en qué posición (índice) del arreglo se encontró.

**Ejercicio 3:**

a. Defina una clase *Tecla* para representar una tecla de un piano. Cada tecla puede estar o no apretada, y tiene además una nota asociada (cuyo nombre se representará con un *string*). Su interfaz debe tener tener entonces:

- un constructor que reciba el nombre de la nota
- un método *VerNota* que retorne el nombre de la nota
- un método *Apretar* que cambie el estado de la tecla a apretada.
- un método *Soltar* que cambie el estado de la tecla a no apretada.
- un método *EstaApretada* que retorne *true* si la tecla está apretada, *false* en caso contrario

b. Defina una clase *Pedal* para representar el pedal de un piano. El pedal debe almacenar un valor (*float*) que indica la presión que el músico ejerce sobre el pedal. El constructor debe inicializar la presión en *0*, y la clase debe tener métodos para modificarla y consultarla.

c. Reutilizando las clases *Tecla*, *Pedal* e *Instrumento*:

```
class Instrumento{
        virtual string VerTipo() { return "sin_nombre"; }
};
```

defina una clase *Piano* que modele un instrumento de tipo *"piano"* con solo 7 teclas (“*do”, “re”, “mi”, “fa”, “sol”, “la” y “si”*) y 1 pedal. La clase piano debe tener métodos para:

- apretar una tecla, indicando el número de tecla, y que retorne la nota que debería sonar.
- soltar una tecla, indicando el número de tecla
- presionar el pedal, indicando la presión que se aplica

**Ejercicio 4 (20 pts)** a) ¿Qué es un objeto? ¿Qué es una clase? b) ¿Cuándo se invoca al constructor de un objeto? ¿Cuándo al destructor? c) ¿Cuáles son las ventajas de utilizar memoria dinámica (con new y delete)? d) ¿Qué significa que una función F sea amiga de una clase C? ¿Cómo se declara dicha amistad? e) Si en una clase no se declara ningún constructor, ¿qué constructores tengo por defecto? ¿en qué casos dichos constructores son incorrectos?

**Parcial 1 2017 Tema A**

**Ejercicio 1 (25 pts)** Implemente una clase Fraccion para representar una fracción. La clase debe tener: a) un constructor que no reciba nada y la inicialice como 0/1, b) un constructor que reciba numerador y denominador, c) métodos para consultar numerador y denominador, d) sobrecargas para los operadores + y -, e) sobrecargas para los operadores == y !=, y f) una sobrecarga para poder mostrar una instancia "f" de la clase mediante "cout << f".

**Ejercicio 2 (25 pts)** Escriba una función llamada max_element(...) que reciba dos punteros a de tipo float, uno indicando el comienzo de un arreglo (dirección del 1er elemento), y otro indicando el final (dirección siguiente del último elemento), y retorne un puntero indicando la posición del mayor elemento del arreglo. Implemente un programa cliente que genere un arreglo de 10 elementos y muestre a qué índice (0...9) corresponde el mayor.

**Ejercicio 3 (30 pts)** a) Escriba una clase Jugador para representar un jugador en un juego de Piedra-Papel-Tijeras. La clase debe poder guardar el nombre del jugador, y tener además un método virtual Jugar() que retorne un entero. El número que retorna puede ser 1 (piedra), 2 (papel) o 3 (tijeras), e indica la jugada que realiza el jugador en una partida.

b) Implemente dos herencias, JugadorIA y JugadorHumano. La primera debe elegir su jugada al azar, y tener siempre por nombre de jugador "HAL 9000". La segunda debe solicitar al usuario que elija la jugada que quiera (nota: deberá utilizar cin/cout dentro de esta segunda clase), y recibir el nombre del humano como argumento en su constructor.

c) Implemente una función que reciba dos Jugadores, simule una partida, y retorne el nombre del ganador, o la palabra "empate".

**Ejercicio 4 (20 pts)** Explique: a) ¿Cómo identifica cuándo debe usar herencia y cuándo composición? b) ¿Qué significa que un método sea "virtual puro"? c) ¿Cuando en una clase es necesario implementar un operador y un constructor de copia? d) ¿Señale ventajas y desventajas de utilizar new y delete para gestionar la memoria? e) ¿Qué es y por qué es importante el "principio de ocultación"?

**Parcial 1 2017 Tema B**

**Ejercicio 1 (pts)** Implemente una clase Complejo para representar un número complejo. La clase debe tener: a) un constructor que no reciba nada y la inicialice como 0+0i, b) un constructor que reciba las partes real e imaginaria, c) métodos para consultar cada parte, d) sobrecargas para los operadores + y -, e) sobrecargas para los operadores == y !=, y f) una sobrecarga para poder mostrar una instancia “c” de la clase mediate “cout << c”.

**Ejercicio 2 (25 pts)** Escriba una función find_if_even(...) que reciba dos punteros a int, uno indicando el comienzo de un arreglo (dirección del 1er elemento), y otro indicando el final (dirección siguiente del último elemento), y retorne un puntero indicando la posición del primer número par que encuentre. Si no encuentra ninguno debe retornar null. Implemente un programa cliente que genere un arreglo de 10 elementos y muestre a qué índice (0...9) corresponde el encontrado (o algún mensaje alusivo si no encuentra ninguno).

**Ejercicio 3 (30 pts)** a) Escriba una clase Promocion para representar una promoción de una tienda virtual. La clase debe poder guardar una descripción de la promoción, y tener además un método virtual Aplicar(...) que reciba un struct de tipo Compra como el del recuadro y retorne un float con el monto que el cliente se ahorra gracias a esa promoción, o cero si la promoción no es aplicable a esa compra.

b) Implemente dos herencias, Promo3x2 y Promo1000. La primera, cuya descripción será "3x2 en todas las marcas" aplica cuando un cliente compra 3 o más unidades de un mismo producto; y en ese caso por cada 3 unidades, se le descuenta una. La segunda, cuya descripción será "5% en compras superiores a $1000" aplica cuando la compra es mayor a $1000, y en ese caso se le descuenta un 5%.

c) Implemente una función que reciba un vector de Compras y dos Promociones, y retorne un vector de Descuentos. La función debe intentar aplicar cada promoción a cada compra, y generar un vector con todos los descuentos de aquellas que efectivamente apliquen.

```
struct Compra {
        string codigo_producto;
        float precio_unitario;
        int cantidad;
};
struct Descuento {
        string descripcion;
        float monto;
};
```

**Ejercicio 4 (20 pts)** Explique: a) ¿Cómo identifica cuándo debe usar herencia y cuándo composición? b) ¿Qué significa que un método sea “virtual puro”? c) ¿Cuando en una clase es necesario implementar un operador y un constructor de copia? d) ¿Señale ventajas y desventajas de utilizar new y delete para gestionar la memoria? e) ¿Qué es y por qué es importante el “principio de ocultación”?

**Recuperatorio 1 2017**

**Ejercicio 1 (25 pts)** a) Escriba una función que reciba 3 punteros de tipo float, el primero y segundo indican el comienzo y final de los elementos que deben ser copiados de un arreglo hacia otro. El comienzo del vector destino es apuntado por el tercer puntero recibido. b) Haga uso de la función desde un programa cliente.

**Ejercicio 2 (25 pts)** a) Escriba una clase Vector3D para representar un vector de 3 componentes {x;y;z}. b) Implemente sobrecargas para los operadores [ ], + y ==. El operador [ ] debe permitir además modificar las componentes. Nota: dados dos vectores {x1;y1;z1} y {x2;y2;z2}, la suma es un vector {x1+x2 ; y1+y2 ; z1+z2}. c) Sobrecargue el operador << para la salida ([x, y, z]) y del operador >> para su lectura.

**Ejercicio 3 (30 pts)** a) Escriba una clase JuegoNaipes para representar un juego de naipes. La clase debe tener como atributo un mazo que es un vector de naipes, representados por un struct que contiene palo (string) y valor (string) y un método virtual denominado generar() b) Sobrecargue el operador [] para que retorne un naipe dada su posición. c) Implemente dos herencias, Truco y BlackJack. El método generar() de Truco debe inicializar el mazo con 40 naipes del 1 al 12 por cada palo (basto, espada, oro y copa) excluyendo los 8 y 9. De modo similar, el de la clase BlackJack debe inicializar el mazo con 52 naipes, del 1 al 13 por cada palo (trébol, diamante, corazón y pica). d) Implemente una función libre (no método) llamada GenerarYMostrar, que reciba un JuegoNaipes, genere el mazo y muestre por consola las cartas generadas.

**Ejercicio 4 (20 pts)** a) Enumere y explique 3 ventajas de utilizar memoria dinámica (new y delete). b) ¿Una clase hija, puede acceder a los atributos o métodos privados de la clase padre? Explique. c) Si en una clase no se declara ningún constructor, ¿qué constructores tengo por defecto? ¿en qué casos dichos constructores son incorrectos? d) ¿Qué significa que una función F sea amiga de una clase C?