**CLASES Y OBJETOS**

# 1. Ejercicio

Dada la clase:

**class** Televisor

{

**public**:

**bool** encendido**;**

**unsigned int** canal**;** *// Válido de 1 a 130 (ambos inclusive)*

**unsigned int** volumen**;** *// Válido de 0 a 100 (ambos inclusive)*

**void** prendeApaga (**void**);

**void** setCanal(**unsigned int** canal);

**void** subirCanal(**void**);

**void** bajarCanal (**void**);

**void** subirVolumen(**void**);

**void** bajarVolumen (**void**);

};

Programar los métodos declarados e implementar un programa de prueba de los mismos.

# 2. Ejercicio

Dada la clase:

**class** Fecha

{

**public**:

**unsigned** **int** anio;

**unsigned** **int** mes;

**unsigned** **int** dia;

**unsigned** **int** hora;

**unsigned** **int** minuto;

**unsigned** **int** segundo;

**std::string** formatoISO(**void**); *// Formato aaaa-mm-dd HH:MM:SS, ej: 1999-03-31 10:08:38*

**bool** esAnterior(Fecha x);

**bool** esIgual(Fecha x);

**bool** esPosterior(Fecha x);

};

Programar los métodos declarados e implementar un programa de prueba de los mismos. Pueden programarse métodos adicionales, si es necesario.

# 3. Ejercicio

Crear una clase llamada Racional que realice operaciones aritméticas con fracciones.

Mediante variables enteras represente los datos de la clase (numerador y denominador). Proporcione funciones miembro públicas para lo siguiente:

* Suma de dos números racionales. El resultado deberá almacenarse en forma reducida.
* Resta de dos números racionales. El resultado deberá almacenarse en forma reducida.
* Multiplicación de dos números racionales. El resultado deberá almacenarse en forma reducida.
* División de dos números racionales. El resultado deberá almacenarse en forma reducida.
* Impresión de números racionales en la forma a/b, donde a es el numerador y b el denominador.
* Impresión de números racionales en formato de punto flotante.

Implementar un programa de prueba.

# 4. Ejercicio

Dadas las clases:

**class** Persona

{

**public**:

**unsigned** **int** edad;

**std**::**string** nombre;

};

**class** Multitud

{

**public**:

Persona lista[1000];

**unsigned** **int** cargados;

**void** inicializar(**void**); *// Descarta lista (cargados = 0)*

**bool** agregar (Persona p); *// Retorna falso si no hay lugar (+1000)*

**std::string** csvMayores(**void**); *// listado csv de nombres de edad >=18*

**std::string** csvMenores(**void**); *// listado csv de nombres de edad < 18*

**void** ordenarEdadCreciente(**void**); *// Ordena lista por edad de menor a mayor*

};

Programar los métodos declarados e implementar un programa de prueba de los mismos. Pueden programarse métodos adicionales, si es necesario.

# 5. Ejercicio

Crear la clase Vendedor que incluya los siguientes datos: nombre, sueldo básico, ventas y comisión. También proporcione el método sueldoACobrar (que calcule y retorne el resultado del sueldo a cobrar por el vendedor teniendo en cuenta el sueldo básico y la comisión que recibe por las ventas realizadas).

Crear una segunda clase llamada Nomina que pueda contener hasta 1000 vendedores y que contenga los siguientes métodos:

* agregarVendedor: incorpora un vendedor a la nómina.
* mejorVendedor: retorna el objeto Vendedor con mayores ventas.
* csvTop5Comisiones: retorna string de los nombres de los 5 (o menos si no hay tal cantidad) vendedores con mejor comisión.
* csvBottom5SueldoTotal: retorna string de los nombres de los 5 (o menos si no hay tal cantidad) vendedores con peor sueldo a cobrar.

Implementar un programa de prueba.

# 6. Ejercicio

Definir la clase Alumno con los siguientes datos miembro: nroDocumento, nombre, nota1, nota2, nota3 y funciones miembro: asignarDatos, promedio, mayorNota, menorNota.

Luego definir una clase Materia que contenga hasta 30 alumnos inscriptos y que permita obtener de los mismos, la siguiente información:

* inscribirAlumno: agrega un alumno a la materia (sin repetir alumnos basado en nroDocumento). Retornar true o false según se haya inscripto al alumno o no.
* desinscribirAlumno: elimina un alumno a la materia (búsqueda basada en nroDocumento). Retornar true o false según se haya desinscripto al alumno o no.
* csvAlumnosAprobados: nombres de los alumnos cuyas 3 notas alcancen o superen los 4 puntos.
* csvTop5Promedios: nombres de los alumnos con los 5 (o menos de no haber suficientes) alumnos con mejor promedio.

Implementar un programa de prueba.

# 7. Ejercicio

Definir las clases:

* Clase "Movimiento". Propiedades: fecha (string en AAAA-MM-DD), monto (positivo para depósitos, negativo para extracciones). Métodos: constructor
* Clase "Cuenta". Propiedades: número de cuenta, nombre del cliente, saldo y movimientos (arreglo de hasta 1000 objetos Movimiento). Métodos: constructor, extraer y depositar (ambas con parámetro monto double), recalcular saldo (reprocesa todos los movimientos para restablecer el saldo) y csv (fecha,monto por movimiento) de los últimos n movimientos (donde n es la cantidad de movimientos más recientes a obtener).

Aplicarlas en un programa de prueba.

# 8. Ejercicio

Cree la clase ConjuntoEnteros. Cada objeto ConjuntoEnteros puede almacenar enteros en el rango de 0 a 10. El conjunto se representa mediante un arreglo de valores boolean. El elemento del arreglo a[i] es true si el entero i se encuentra en el conjunto. El elemento del arreglo a[j] es false si el entero j no se encuentra dentro del conjunto. Inicialmente todos los valores del arreglo están en false.

Proporcione los siguientes métodos:

* el método unión debe crear un tercer conjunto que sea la unión teórica de conjuntos para los dos conjuntos existentes (es decir, un elemento del tercer arreglo se establece en true si ese elemento es true en cualquiera o en ambos de los conjuntos existentes; en caso contrario, el elemento del tercer conjunto se establece en false).
* El método intersección debe crear un tercer conjunto que sea la intersección teórica de conjuntos para los dos conjuntos existentes (es decir, un elemento del arreglo del tercer conjunto se establece en false si ese elemento es false en uno o ambos de los conjuntos existentes; en caso contrario, el elemento del tercer conjunto se establece en true).
* El método insertarElemento debe insertar un nuevo entero k en un conjunto (estableciendo a[k] en true).
* El método eliminarElemento debe eliminar el entero m (estableciendo a[m] en false).
* El método mostrarConjunto debe imprimir el conjunto como una lista de números separados por espacios. Incluya sólo aquellos elementos que estén presentes en el conjunto. Use "{ }" para representar un conjunto vacío.
* El método esIgualA debe determinar si dos conjuntos son iguales.

Escriba un programa para probar la clase ConjuntoEnteros. Cree dos instancias de objetos ConjuntoEnteros. Implementar un programa de prueba.