B.- Ejercicio de Herencia y Polimorfismo: FORMAS

1. Defina una clase `Forma` que tenga los siguientes miembros de datos:
   - Color
   - Coordenada del centro de la forma (objeto Punto)
   - Nombre de la forma (`char *`)

   Y, al menos, las siguientes funciones miembro:
   - Imprimir
   - Obtener y cambiar el color
   - Mover la forma (o sea, su centro)

   Defina una clase derivada `Rectangulo` que tenga los siguientes miembros como datos:
   - Lado menor.
   - Lado mayor.

   Y, al menos, las siguientes funciones miembro:
   - Imprimir. Debe imprimir qué se trata de un rectángulo mostrando su nombre, color, centro y lado. Debería usarse la función Imprimir de la clase base para realizar parte de este trabajo.
   - Calcular el área (lado menor * lado mayor).
   - Calcular el perímetro. (2 * lado menor + 2 * lado mayor).
   - Cambiar el tamaño del rectángulo. Recibe como parámetro un factor de escala. Así, por ejemplo, si el factor vale 2, el rectángulo duplicará su tamaño y si es 0,5 se reducirá a la mitad.

   Realice un programa que pruebe el funcionamiento de estas clases. Debe crear objetos y comprobar el correcto funcionamiento de las funciones miembro.
2. Defina una clase `Elipse` derivada de forma. Recordatorio: una elipse queda definida por su radio mayor (R) y su radio menor (r), tal que el área de una elipse es igual a $\pi*(R*r)$.
3. Defina una clase `Cuadrado` derivada de la clase `Rectangulo`.
4. Defina una clase `Circulo` derivada de la clase `Elipse`.
5. Realice un programa que defina varias formas diferentes, cree un vector de punteros de la clase `Forma` que apunten a los objetos creados. El programa debe realizar un bucle que recorra todas las formas, las ponga todas del mismo color y las mueva a una determinada posición.
6. Analice qué ocurre en el ejercicio anterior si se intenta imprimir la información de cada forma y qué sucede si se intenta obtener en ese bucle el área de todas las formas del vector.
7. Utilice la técnica de las funciones virtuales para arreglar los comportamientos anómalos detectados en el ejercicio anterior.
8. Desarrolle un programa que, dado un conjunto de formas, calcule cuál tiene el área máxima e imprima la información de dicha forma.
9. ¿Cómo haría para obligar que todas las clases futuras derivadas tengan al menos los métodos “área” y “perímetro”. ¿Tiene sentido incluir definir dichos métodos en la clase “forma”?
10. Haga un diagrama de clases que refleja la estructura definida hasta el momento. Añádale las clases: Punto, Línea, Triángulo, Triángulo Rectángulo y Polígono. ¿Dónde irían? ¿Cuáles serían sus atributos y propiedades? (No los implemente).

C.- Ejercicio de Herencia y Polimorfismo: EMPLEADOS

Se pretende desarrollar un conjunto de clases que representen, de forma simplificada, a una hipotética empresa dedicada a vender un producto. A continuación, se describen las características básicas de estas clases:

1. `Empleado`. Clase básica que describe a un empleado. Incluye sus datos personales (nombre, apellidos, DNI, dirección) y algunos datos tales como los años de antigüedad, teléfono de contacto y su salario.
   Incluye también información de quién es el empleado que lo supervisa (`Empleado *`). Tendrá, al menos, las siguientes funciones miembro:

   - Constructores para definir correctamente un empleado, a partir de su nombre, apellidos, DNI, dirección, teléfono y salario.
   - Imprimir (A través de los operadores de E/S redefinidos)
   - Cambiar supervisor
   - Incrementar salario

2. `Secretario.` Tiene despacho, número de fax e incrementa su salario un 5% anual.
   Tendrá, al menos, las siguientes funciones miembro:

   - Constructores (debe rellenar la información personal y los datos principales)
   - Imprimir (debe imprimir sus datos personales y su puesto en la empresa).

3. `Vendedor.` Tiene coche de la empresa (identificado por la matricula, marca y modelo), teléfono móvil, área de venta, lista de clientes y porcentaje que se lleva de las ventas en concepto de comisiones. Incrementa su salario un 10% anual.
   Tendrá, al menos, las siguientes funciones miembro:

   - Constructores (debe rellenar la información personal y los datos principales)
   - Imprimir (debe imprimir sus datos personales y su puesto en la empresa).
   - Dar de alta un nuevo cliente.
   - Dar de baja un cliente.
   - Cambiar de coche.

4. `Jefe de zona.` Tiene despacho, tiene un secretario a su cargo, una lista de vendedores a su cargo y tiene coche de la empresa (identificado por la matrícula, marca y modelo). Incrementa su salario un 20% anual.
   Tendrá, al menos, las siguientes funciones miembro:
   - Constructores (debe rellenar la información personal y los datos principales)
   - Imprimir (debe imprimir sus datos personales y su puesto en la empresa).
   - Cambiar de secretario.
   - Cambiar de coche.
   - Dar de alta y de baja un nuevo vendedor en su zona.

Todos los empleados son vendedores, jefes de zona o secretarios. Hacer un programa de prueba que muestre como funciona. Probar, especialmente, que el método incrementar salario se comparta bien, según el empleado que sea así es la subida.

D.- Ejercicio de herencia y polimorfismo: Gestión Facultad

1. Se pretende realizar una aplicación para esta facultad que gestione la información sobre las personas vinculadas con la misma, que se pueden clasificar en tres tipos: estudiantes, profesores y personal de servicio.

   A continuación, se detalla qué tipo de información debe gestionar esta aplicación:

   - Por cada persona, se debe conocer, al menos, su nombre y apellidos, su número de identificación y su estado civil.
   - Con respecto a los empleados, sean del tipo que sean, hay que saber su año de incorporación a la facultad y qué número de despacho tienen asignado.
   - En cuanto a los estudiantes, se requiere almacenar el curso en el que están matriculados.
   - Por lo que se refiere a los profesores, es necesario gestionar a qué departamento pertenecen (lenguajes, matemáticas, arquitectura, ...).
   - Sobre el personal de servicio, hay que conocer a qué sección están asignados (biblioteca, decanato, secretaría, ...).

   El ejercicio consiste, en primer lugar, en definir la jerarquía de clases de esta aplicación. A continuación, debe programar las clases definidas en las que, además de los constructores, hay que desarrollar los métodos correspondientes a las siguientes acciones:

   - Cambio del estado civil de una persona.
   - Reasignación de despacho a un empleado.
   - Matriculación de un estudiante en un nuevo curso.
   - Cambio de departamento de un profesor.
   - Traslado de sección de un empleado del personal de servicio.
   - Imprimir toda la información de cada tipo de individuo.

   Incluya un programa de prueba que instancie objetos de los distintos tipos y pruebe los métodos desarrollados.

2. Se plantea extender el ejemplo anterior incluyendo una clase que represente al centro docente. Esa clase incluirá 3 contenedores, uno por cada tipo de persona vinculada con el centro.

   En una primera fase deben incluirse los siguiente métodos:

   - Uno para dar de alta una persona, que incoporará a la persona en la lista correspondiente.
   - Otro para dar de baja una persona, dado su DNI. Añada un método a la clase persona para poder obtener el DNI de un objeto de esa clase.
   - Uno último para imprimir toda la información de las personas vinculadas con el centro.

Dado que hay múltiples altenativas a la hora de afrontar este problema, a continuación, se explican las características de la solución implementada, aunque el alumno podrá realizar el diseño que considere oportuno:

- Se usan como contenedores listas de punteros a estudiantes, profesores y personal de servicio, respectivamente.
- El método de alta recibe como parámetro un objeto del tipo persona correspondiente, existiendo, por tanto, tres versiones sobrecargadas del mismo.
- Cada versión del método crea un objeto dinámico (`new`) del tipo que le corresponde y lo inserta en la lista.
- El método de baja busca, en primer lugar, un objeto en las listas que tenga el DNI recibido como parámetro. Una vez encontrado, lo libera (`delete`) y lo elimina de la lista (`erase`).

3. **Nota** previa: Guarde la versión anterior y realice este ejercicio sobre una copia de la misma, puesto que dicha versión previa será utilizada en ejercicios posteriores.

   En la aplicación anterior, se detecta que en el futuro se va a necesitar crear nuevos tipos de empleados (por ejemplo, investigadores) y distinguir entre distintos tipos de estudiantes. Por tanto, para que el diseño se pueda adaptar a estas necesidades futuras, se plantea unificar todos los contenedores del tipo centro, de manera que sólo haya un único contenedor de punteros a personas, y usar polimorfismo para gestionar los distintos tipos de personas.

   Modifique el programa anterior para adaptarlo a este nuevo diseño. Tenga en cuenta que debe desaparecer cualquier referencia a profesores, personal de servicio y estudiantes en el código de la clase que representa al centro. Asimismo, haga que las clases que corresponden a personas y empleados sean abstractas.

4. En la versión correspondiente al tercer ejercicio, se va a añadir un nuevo método a las clases profesor y personal de servicio para calcular su salario. Se va a suponer que la forma de calcular el salario para estos dos tipos de empleados es totalmente distinta y que no se puede sacar ningún código común para incluirlo en la clase empleado. Haga la suposición que considere oportuna sobre cómo calcular el sueldo de cada tipo de empleado, dado que no es importante para el ejemplo. Aquí va una propuesta bastante ridícula: el personal de servicio cobra una cantidad fija más un 5% si están casados; el personal docente gana un fijo más un 8% si su fecha de incorporación es anterior al 2000. Habilite los métodos en la clase base que necesite para obtener los datos que requiera para el cálculo del sueldo.

   Una vez añadidos los métodos para calcular el salario en cada una de las dos clases, debe añadir un método en la clase que representa el centro que imprima el salario de todos los empleados del centro (nombre y apellidos más el sueldo).

5. Incorpore la funcionalidad del ejercicio previo a la versión polimórfica del cuarto ejercicio. Para ello, se plantea que la clase centro gestione, además del contenedor de punteros a persona, uno adicional de punteros a empleados. De esta forma, al dar de alta a un empleado se insertará un puntero al mismo en ambas listas y al darlo de baja se le eliminará también de ambas.

6. Se plantea mejorar las clases diseñadas hasta ahora en los siguientes aspectos:

    - Añadir una operación `leer` en cada clase persona para leer los datos de la clase.
    - Sobrecargar el operador `cout` para que imprima los datos de cada clase persona.
    - Sobrecargar el operador `cin` para que lea los datos de cada clase persona.
    - Sobrecargar el operador `cout` para que imprima los datos del centro.

7. Hagamos algunas mejoras adicionales en la aplicación desarrollada:

    - Hacer que el programa principal dialoge con el usuario pidiéndolo qué tipo de operación quiere llevar a cabo (alta, baja, imprimir listado o imprimir nóminas) y le vaya pidiendo los datos que necesite para llevar a cabo la operación.
    - Reorganizar el código de manera que cada clase esté en un fichero separado, dejando la implementación de cada clase en ficheros con extensión `cpp` y la definición en ficheros con extensión `h`.