Deusto
Facultad de Ingenieria

EXAMEN PROGRAMACIÓN IV

(Final C/C++, 07 de junio de 2016)

Nombre y apellidos: _____

Código individual: _____

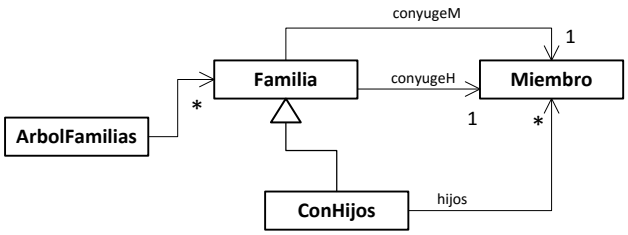
INSTRUCCIONES PARA EL EXAMEN

- La entrega del examen resuelto se realiza a través de la plataforma ALUD2. Existe una tarea para la entrega del examen llamada **Entrega examen 07/06/2016**
- El alumno debe crear un único **archivo ZIP** que contenga TODO el proyecto C++ y los ficheros requeridos, y subirlo a la plataforma. Solamente se corregirán los archivos entregados. Comprobar el contenido del ZIP.
- Es imprescindible que en el fichero correspondiente al programa principal se incluya en la parte superior, como comentario, el nombre del alumno y el **código individual** que se entregará en el examen.
- La duración del examen es de **2 horas**.
- Se puede hacer uso de la referencia de la librería de C/C++ contenida en ALUD2
- Se permite el uso de cualquier material en el examen (prácticas y apuntes).

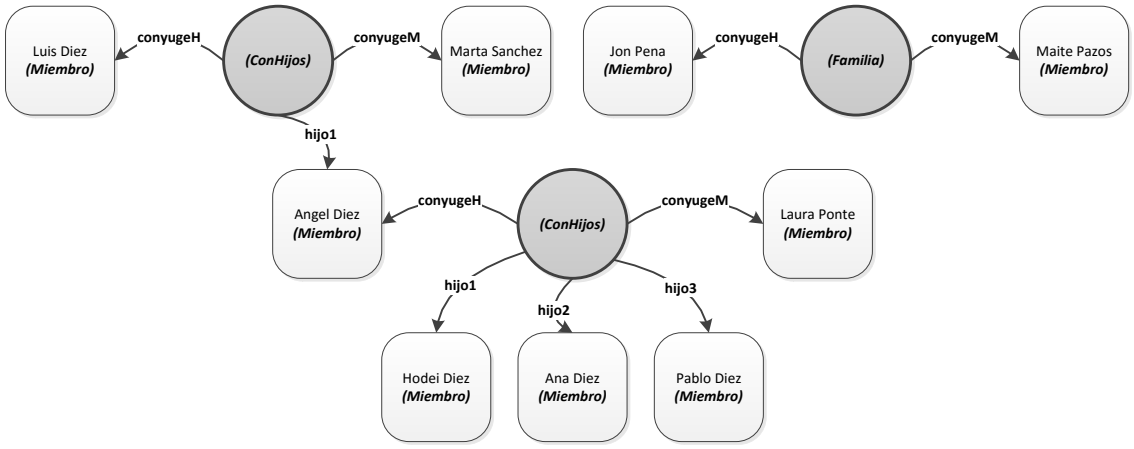
0. LOS PRELIMINARES…

El examen consistirá en implementar las clases necesarias para representar en un grafo la información relativa a los miembros de una serie de familias y sus parentescos (cónyuges, padres, madres e hijos). Para ello se parte de la clase **`Miembro`** ya creada (con sus correspondientes ficheros de cabecera **.h** y de implementación **.cpp**) y un fichero **"main.cpp"** que contiene el programa principal. Este programa crea e inicializa 9 miembros, les asigna un nombre y una edad e imprime por pantalla los datos de uno de ellos. Antes de comenzar el examen, prueba que este mínimo programa funciona correctamente de modo que estés seguro que el entorno de desarrollo se encuentra bien configurado.

Durante la fase de diseño del programa se llegó a la conclusión de que la casuística de posibles familias y operaciones requiere de las clases **`Miembro`**, **`Familia`**, **`ConHijos`** y **`ArbolFamilias`** con estas relaciones:



Seguidamente se presenta el grafo de instancias con 3 familias, sus miembros y parentescos al que nos referiremos después en el examen. No debes comprenderlo en este punto, lo irás haciendo más adelante.



NOTA: los parámetros indicados como `<TYPE>` a lo largo de este enunciado para las distintas funciones a implementar son orientativos, es decir, no se detalla en ningún caso si éstos deben ser punteros, clases, colecciones,... Será el alumno quien deba decidir y actuar en consecuencia.

1. CLASE FAMILIA (0,5 puntos)

1.1 Familia. Crea una nueva clase **`Familia`** (con su fichero de cabecera y de implementación) que represente a un matrimonio entre un hombre y una mujer (cónyuges).

1.2 Atributos de la clase. Esta clase deberá contener los vínculos al par de cónyuges que forman la familia (deberás hacer uso de la clase **`Miembro`** proporcionada). Ten en cuenta el hecho de que pudiera haber miembros, como es el caso de "Angel Diez" que pertenezcan a diferentes familias, desempeñando roles diferentes (hijo de sus padres y marido de su cónyuge).

1.3 Métodos de la clase. Añade en la clase **`Familia`**, al menos, los siguientes métodos:

> **`constructores y métodos get/set`**: incluye aquellos básicos que vayas viendo que necesitas durante el examen para la inicialización y acceso a las instancias de esta clase.
>
> **`char* getNombre()`**: devuelve una cadena de caracteres que concatena (usando un guion como separador) los apellidos del hombre y de la mujer. Por ejemplo "Diez - Sanchez".
>
> En este punto deberás demostrar tus destrezas trabajando con cadenas de caracteres. Para simplificar, consideramos que en todos los casos el apellido de un miembro se encuentra a continuación del primer espacio en blanco de su nombre.
>
> **`void imprimir()`**: su cometido es visualizar por pantalla el nombre de la familia (usar el método anterior) y el nombre completo y edad del marido y de la mujer. En el apartado 3 tienes un ejemplo del formato que se espera en la visualización de esta información.

2. CLASE CONHIJOS (0,5 puntos)

2.1 ConHijos. Crea una nueva clase **`ConHijos`** (con su fichero de cabecera y de implementación) que represente a una familia con descendencia. Esta clase deberá **heredar** de la clase **`Familia`**.

2.2 Atributos de la clase. Esta clase, además de los atributos heredados, deberá incluir el conjunto de hijos de esa familia (deberás hacer uso de la clase **`Miembro`** proporcionada). Ten presente que el número de hijos no es un dato conocido a priori y que variará de unas familias a otras.

2.3 Métodos de la clase. Añade en la clase **`ConHijos`**, al menos, los siguientes métodos:

> **`constructores y métodos get/set:`** incluye aquellos básicos que vayas viendo que necesitas durante el examen para la inicialización y acceso a las instancias de esta clase.
>
> **`void imprimir()`**: debe redefinir el método correspondiente de la clase **`Familia`**. Su cometido es visualizar por pantalla el nombre de la familia, el de sus cónyuges (con su edad) y

los nombres y edad de cada uno de sus hijos. En el apartado 3 tienes un ejemplo del formato que se espera en la visualización de esta información. Se espera que reutilices la funcionalidad común que está implementada en el método equivalente de la clase padre.

3. PROGRAMA PRINCIPAL #1 (0,5 puntos)

Partir del fichero **"main.cpp"** proporcionado, el cual crea inicialmente 9 instancias de la clase **`Miembro`**.

Atención: NO se puede modificar el código que ya se proporciona implementado en este fichero, sí puedes añadir nuevo código, pero las líneas proporcionadas deben persistir invariadas.

3.1 Configurar el grafo. Usando las clases **`Familia`** y **`ConHijos`** que implementaste anteriormente, crea las instancias necesarias para representar la información de las 3 familias que aparecen en el grafo que aparece al principio de este enunciado.

3.2 Visualizar por pantalla dos familias del grafo. Usando el método imprimir de cada clase, visualizar la información de una de las familias sin hijos (Pena - Pazos) y la de otra de las familias con hijos (Diez - Ponte). Se debería visualizar el siguiente resultado:

```
---
FAMILIA: Pena - Pazos
Marido: Jon Pena (40 años)
Mujer: Maite Pazos (35 años)
---

---
FAMILIA: Diez - Ponte
Marido: Angel Diez (32 años)
Mujer: Laura Ponte (30 años)

HIJOS:
Hijo 1: Hodei Diez (2 años)
Hijo 2: Ana Diez (7 años)
Hijo 3: Pablo Diez (10 años)
---
```

Atención: el orden en que se visualizan los hijos por pantalla es irrelevante.

4. CLASE ARBOLFAMILIAS (1,25 puntos)

4.1 ArbolFamilias. Crea una nueva clase **`ArbolFamilias`** (con su fichero de cabecera y de implementación) que represente a un árbol genealógico con un conjunto de familias de cualquier tipo (con hijos o sin ellos).

4.2 Atributos de la clase. Esta clase deberá contener el conjunto de familias que forman el árbol (genealógico). Por simplificar, consideramos que el número máximo de familias a incluir en un árbol es 5.

4.3 Métodos de la clase. Añade en la clase **`ArbolFamilias`**, al menos, los siguientes métodos:

`constructores:` incluye aquellos que necesites durante el examen para la inicialización de esta clase.

`void anadirFamilia(<Familia> f)`: añade una nueva familia al árbol genealógico.

`void imprimir()`: visualiza por pantalla la información de cada una de las familias del árbol. Se espera que uses los métodos implementados para este fin en cada una de las familias.

`int contarMenoresEdad(int edad)`:devuelve cuántos miembros (cónyuges e hijos) de las familias del árbol genealógico tienen una edad menor que la indicada como primer parámetro. **Ayuda:** ten en cuenta que la forma en la que realizar el recuento de miembros no es igual en una familia sin hijos que en una con hijos.

`<listado-miembros> getConyugesInicial(<numero-miembros>, char inicial)`: este método devuelve el conjunto de cónyuges (miembros) de las distintas familias del árbol cuya primera letra del nombre coincide con el carácter pasado como segundo parámetro. Además, el método deja en la variable pasada como primer parámetro el número de miembros devuelto. Deberás decidir tú qué tipos de datos exactos deben ser **`<listado-miembros>`** y **`<numero-miembros>`** para conseguir lo que se espera de este método.

5. PROGRAMA PRINCIPAL #2 (0,25 puntos)

Partir del fichero **"main.cpp"**, donde ya tenías 3 familias creadas.

5.1 Crear un árbol de familias y visualizarlo. Crear una instancia de la clase **`ArbolFamilias`** que contenga las 3 familias que ya teníamos creadas. Posteriormente visualizar la información de todas las familias del árbol usando el método imprimir.

5.2 Invocar el resto de métodos implementados en el árbol de familias. Sobre la instancia de la clase **`ArbolFamilias`**, invoca los métodos que te permiten saber cuántos miembros menores de 35 años hay y quiénes son los cónyuges cuyo nombre empieza por la letra 'L'. Visualiza por pantalla los resultados:

```
Menores 35 años: 6 miembros

Conyuges cuyo nombre empieza por L:
Laura Ponte (30 años)
Luis Diez (60 años)
```