Programación Avanzada

LABORATORIO 0

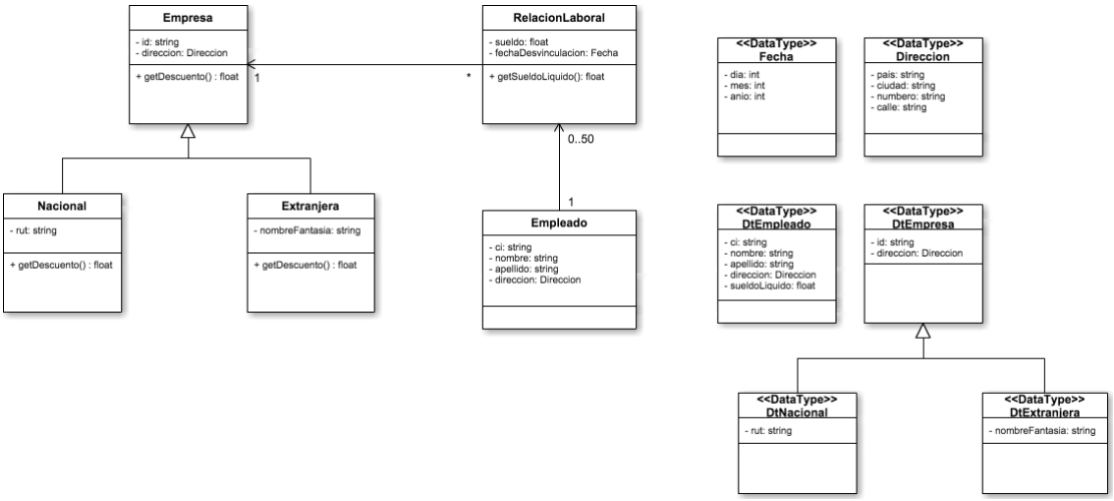
Consideraciones generales:

- La entrega podrá realizarse hasta el **Jueves 4 de abril hasta las 23:59hs.**
- El código fuente y el archivo Makefile deberán ser entregados mediante el moodle del curso dentro de un archivo con nombre <número de grupo>_lab0.zip (o tar.gz). Dentro del mismo archivo comprimido, se deberá agregar un archivo denominado resp_lab0.txt
- El archivo Makefile entregado debe ser independiente de cualquier entorno de desarrollo integrado (*IDE,* por sus siglas en inglés).

Con este laboratorio se espera que el estudiante adquiera competencias en la implementación de operaciones básicas, el uso básico del lenguaje C++ (que se usará en el laboratorio) y, así como reafirmar conceptos presentados en el curso.

Problema

Se desea implementar un pequeño sistema con la realidad expresada en el siguiente modelo, con el fin de manejar información laboral de los empleados.



De los empleados interesa conocer cédula de identidad (que lo identifica), nombre, apellido, domicilio, empresas donde es empleado y el sueldo que cobra en cada empresa. En caso de que el empleado no trabaje más en una empresa se debe guardar la información de que trabajó en ese lugar, y la fecha en que se desvinculó. Si un empleado se desvincula de una empresa, no podrá volver a trabajar en ella. De cada empresa interesa conocer un identificador (cadena de caracteres que la identifica) y la dirección. Las empresas pueden ser nacionales o extranjeras. En caso de ser extranjeras interesa conocer el nombre de fantasía, y en caso de ser nacionales interesa el número de RUT. El sistema debe permitir calcular el salario líquido, que deberá ser obtenido con la operación **`getSueldoLiquido()`**.El mismo se obtiene aplicándole al salario neto, almacenado en el atributo sueldo de cada objeto **`RelacionLaboral`**, un descuento del 35% en caso de trabajar para una empresa nacional, y 20% en caso de trabajar para una empresa extrajera. El porcentaje del descuento deberá ser obtenido utilizando la operación **`getDescuento()`**.

Ejercicio 1

Implementar en C++.

1. Todas las clases (incluyendo sus atributos, pseudoatributos, *getters*, *setters*, constructores y destructores), enumerados y *datatypes* que aparecen en el diagrama. Para las fechas en caso de recibir **dd > 31** o **dd < 1** o **mm > 12** o **mm < 1** o **aaaa < 1900**, se debe levantar la excepción ***std::invalid_argument***. No se deben hacer más que estos controles en la fecha (ej. La fecha 31/2/2000 es válida para esta realidad).

2. Una función ***main*** que implemente las siguientes operaciones:

   a) **`void agregarEmpleado(string ci, string nombre, string apellido, Direccion dir)`** Crea un nuevo empleado en el sistema. En caso de ya existir, levanta la excepción *std::invalid_argument.*

   b) **`void agregarEmpresa(DtEmpresa * empresa)`**, Crea una nueva empresa en el sistema. En caso de ya existir, levanta una excepción *std::invalid_argument.*

   c) **`DtEmpleado** listarEmpleados(int & cantEmpleados)`** Retorna un arreglo de `DtEmpleado*` con todos los empleados del sistema. El largo del arreglo de empleados está dado por el parámetro `cantEmpleados`.

   d) **`void agregarRelacionLaboral(String ciEmpleado, string idEmpresa, float sueldo)`** Vincula un empleado con una empresa. Si la empresa ya está dentro de las empresas que el empleado ha trabajado o trabaja se levanta una excepción *std::invalid_argument*.

   e) **`void finalizarRelacionLaboral(string ciEmpleado, string idEmpresa, Fecha desvinculación)`** Desvincula al empleado de la empresa, registrando la fecha en que terminó el vinculo.

   f) **`DtEmpresa** obtenerInfoEmpresaPorEmpleado(string ciEmpleado, int & cantEmpresas)`** Retorna un arreglo con las empresas donde trabaja activamente el empleado. El largo del arreglo de empresas está dado por el parámetro `cantEmpresas`.

Nota: A los efectos de este laboratorio, la función *main* mantendrá una colección de empleados, implementada como un arreglo de tamaño **MAX_EMPLEADOS**. Lo mismo para las empresas del sistema con un máximo de **MAX_EMPRESAS** (ambas constantes). Puede implementar operaciones en las clases dadas en el modelo si considera que le facilitan para la resolución de las operaciones pedidas en el *main*.

Sobrecargar el operador de inserción de flujo (ej. <<) en un objeto de tipo ***std::ostream***. Este operador debe "imprimir" las distintas clases de **DtEmpresa (DtEmpresaNacional, DtEmpresaExtranjera)** con el siguiente formato:

Id Empresa:

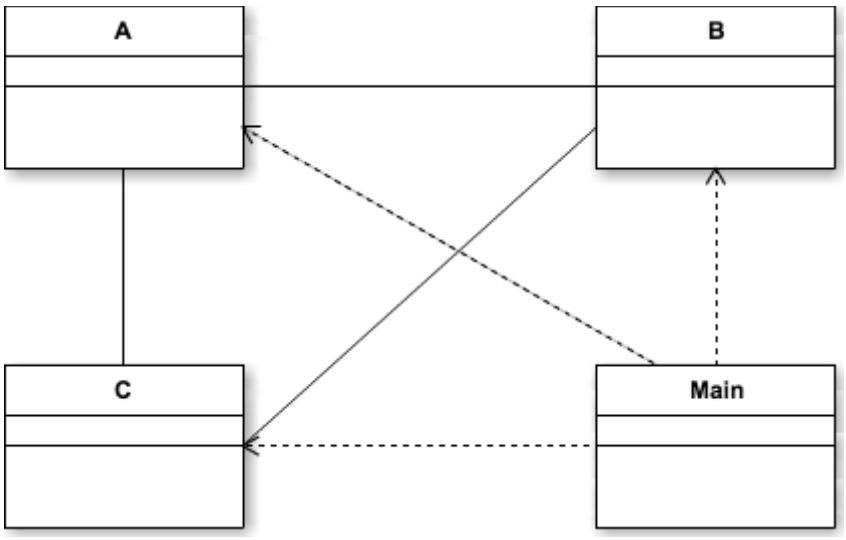
Dirección:

Nombre fantasía o RUT, según corresponda

Notas: El *main* debe manejar las excepciones lanzadas por las operaciones de los objetos.

Ejercicio 2

En este ejercicio se busca que se familiarice con la problemática de dependencias circulares en C++. Para esto se debe implementar y compilar la siguiente estructura dada por el diagrama de abajo.



Preguntas

1. ¿Cuáles son las dependencias circulares que fueron necesarias solucionar para que el programa compile?
2. ¿Qué es *forward declaration*?