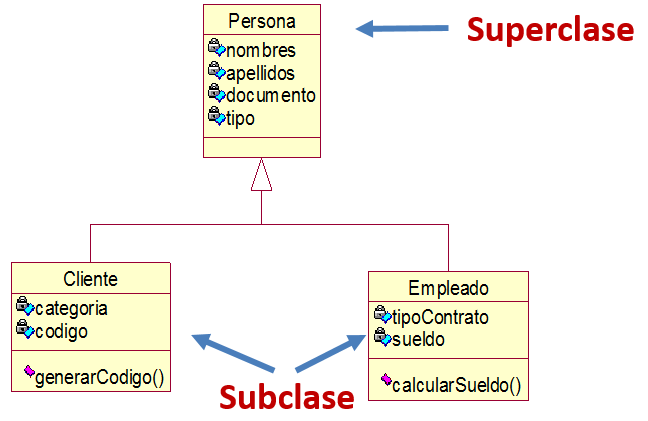
Herencia.

La herencia en programación, es la capacidad de crear clases que pertenezcan al ámbito de otra clase y por lo tanto, puedas reutilizar atributos y métodos de una clase a otra. La herencia te permite establecer una jerarquía a tus clases y definir cuáles atributos y métodos deseas pasar. La clase que está en el punto más alto de esta jerarquía se puede denominar “clase madre” o “superclase” y las que heredan sus atributos y métodos se pueden llamar “subclases”.



**Terminología importante:**

* **Superclase:** la clase cuyas características se heredan se conoce como superclase (o una clase base o una clase principal).
* **Subclase:** la clase que hereda la otra clase se conoce como subclase (o una clase derivada, clase extendida o clase hija). La subclase puede agregar sus propios campos y métodos, además de los campos y métodos de la superclase.
* **Reutilización:** la herencia respalda el concepto de “reutilización”, es decir, cuando queremos crear una clase nueva y ya hay una clase que incluye parte del código que queremos, podemos derivar nuestra nueva clase de la clase existente. Al hacer esto, estamos reutilizando los campos/atributos y métodos de la clase existente.

Sintaxis:

Cuando se declara una clase D derivada de otra clase-base B, se utiliza la siguiente sintaxis:

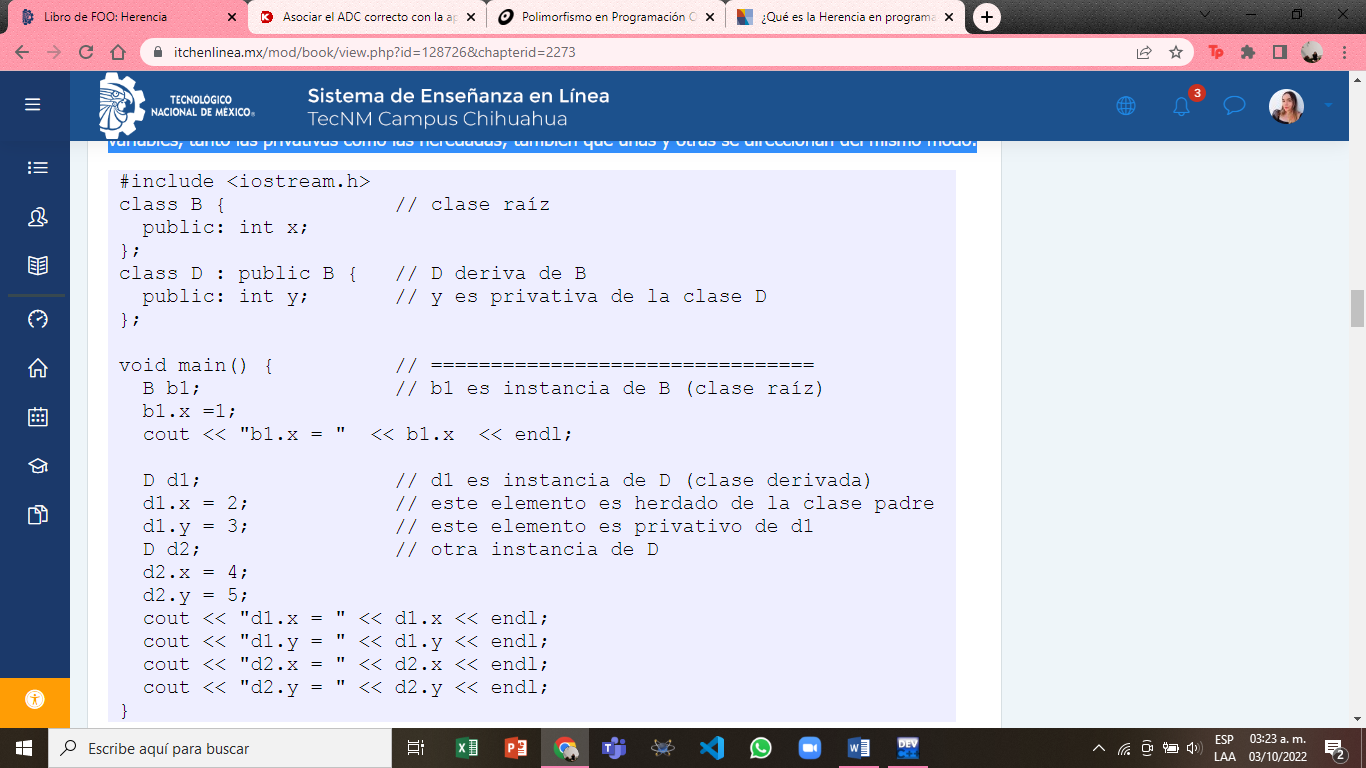
class-key nomb-clase : <mod-acceso> clase-base {<lista-miembros>};

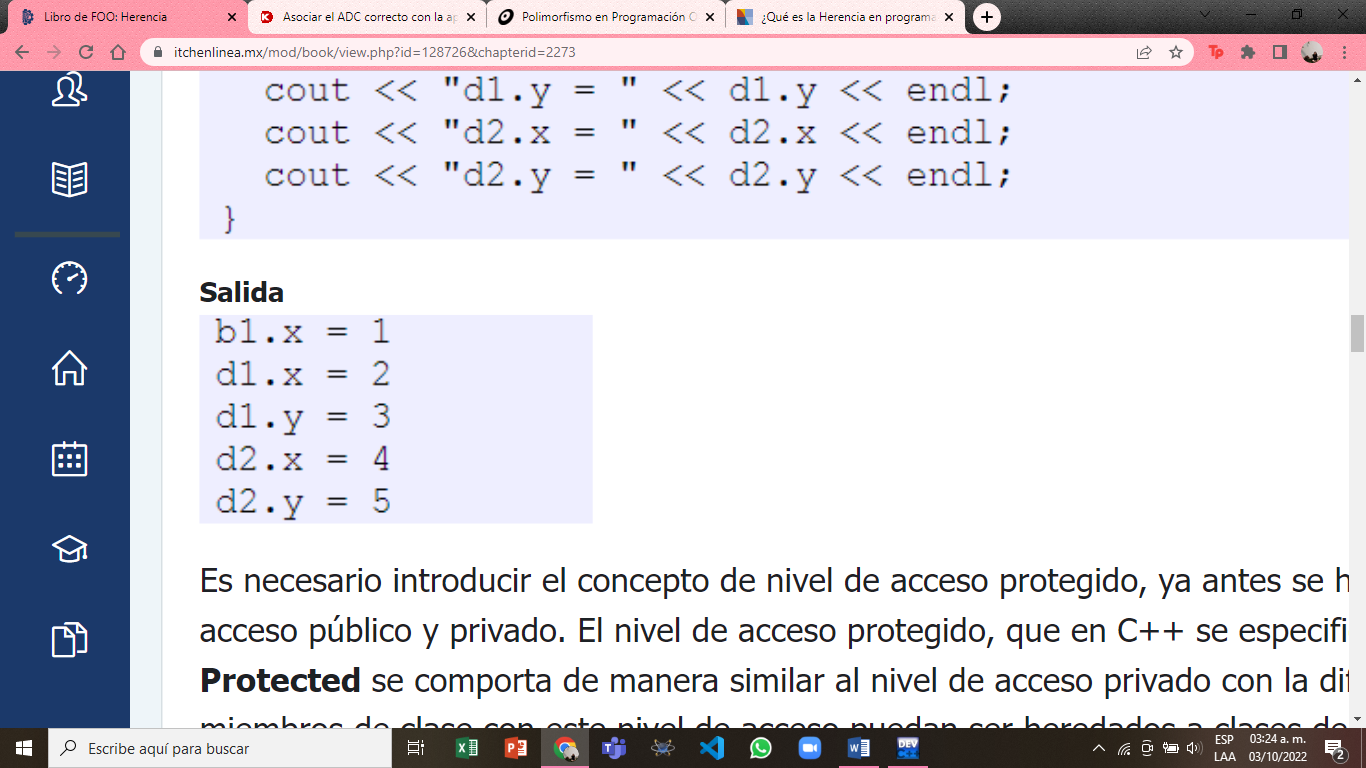
<mod-acceso> es un especificador opcional denominado modificador de acceso, que determina como será la accesibilidad de los miembros que se heredan de la clase-base en los objetos de las subclases.

El significado del resto de miembros es el que se indicó al tratar de la declaración de una clase, con la particularidad que, en la herencia simple, <: lista-base> se reduce a un solo identificador: clase-base.

Ejemplo

En este ejemplo puede comprobarse que cada instancia de la clase derivada tiene su propio juego de variables, tanto las privativas como las heredadas; también que unas y otras se direccionan del mismo modo.





Es necesario introducir el concepto de nivel de acceso protegido, ya antes se había revisado los niveles de acceso público y privado. El nivel de acceso protegido, que en C++ se especifica con la palabra Protected se comporta de manera similar al nivel de acceso privado con la diferencia de que permite que los miembros de clase con este nivel de acceso puedan ser heredados a clases derivadas. Es importante aclarar que los miembros de una clase con acceso privado NO son heredables, como tampoco lo son los constructores, destructores, clases y funciones amigas, y operadores sobrecargados.