Recomendaciones importantes para implementar los TADs en C++

- **Convenios de notación de las clases**
  - Nombres de los atributos propios de la clase
    - Primera posibilidad
      Tipo _atributo1;
      Tipo _atributo2;
    - Segunda posibilidad
      Tipo atributo1_;
      Tipo atributo2_;
  - Observadores y modificadores
    - Primera posibilidad
      Tipo getAtributo(); // Observador
      void setAtributo(Tipo nuevoValor); // Modificador
    - Segunda posibilidad
      Tipo verAtributo(); // Observador
      void cambiarAtributo(Tipo nuevoValor); // Modificador
    - Tercera posibilidad
      Tipo atributo(); // Observador
      void atributo(Tipo nuevoValor); // Modificador

  - Observación
    Se puede elegir la notación que se desee, pero se recomienda que se mantenga su uso durante todo el curso.

- Los observadores se implementan como métodos ***const***.
  - Ejemplo:
    double Point2D::getX() ***const****;*
    double Point2D::getY() ***const****;*

- En general, los parámetros a métodos se pasan como referencias constantes.
  - Ejemplo:
    addPoint(Point2D ***const****& p)*

- Uso de asertos
  - Para implementar las pre/post condiciones e invariantes, se debe usar la macro ***assert*** importada con
    #include <cassert>
    siguiendo el esquema[1]:

```
metodo_constructor(parámetros)
{
  [<precondiciones]
    ...código....
  [<invariantes>]
  [<postcondiciones>]
}

metodo_observador(parametros) const
{
```

1 Lo indicado entre [] es dependiente del caso y puede no haya que reflejarlo.

```
 [<precondiciones>]
 [<invariantes>]
 .... código ...
 //no debería ser necesario comprobar aquí los posibles invariantes
 //ya que no se debe alterar el estado del objeto en un observador.
[postcondiciones]
}

metodo_modificador(parametros)
{
  [<precondiciones>]
  [<invariantes>]
   .... código ... //pueden violarse los invariantes aquí durante la ejecución.
  [<invariantes>]
  [postcondiciones]
}
```

- Ejemplo

```
Line2D::crossPoint(Line2D const& s)
{
 //precondición
 assert(acuteAngle(s) > 0.0)

  //código.

   //Post-condiciones: el punto pertenece a las dos rectas.
   assert( x(retV.y())==s.x(retV.y());
   assert( y(retV.x())==s.y(retV.y());
 }
```

- **Compilación condicional**
  - Cuando se desee referir al estado antiguo del TAD en una post-condición, se debe usar la compilación condicional con la macro ***NDEBUG***.
  - Si esta macro no está definida entonces las sentencias ***assert()*** se aplican.
  - Si se usa '***-DNDEBUG***' como opción de compilación entonces las sentencias ***assert()*** se desactivan.
  - Ejemplo: se va utilizar esta macro para tener código con compilación condicional

```
Point2D::addPoint(Point2D const& p)
{
 // Si no se ha usado -DNDEBUG al compilar
 // entonces se activa la comprobación de asertos.
  #ifndef NDEBUG
      float oldX=getX();
      float oldY=getY();
  #endif

   // Código.

  //post-condiciones.
      assert(getX() == oldX + p.getX());
      assert(getY() == oldY + p.getY());
}
```