Universidad de Córdoba
Escuela Politécnica Superior de Córdoba

ESTRUCTURAS DE DATOS

GRADO EN INGENIERÍA INFORMÁTICA
Segundo curso. Segundo cuatrimestre.
Curso académico 2018 – 2019

PRIMERA PRÁCTICA

MONOMIOS DE UNA VARIABLE

- **OBJETIVO**
  - **Codificar en C++**
    - El tipo abstracto de datos **Monomio**, compuesto por coeficiente y grado
      - *coeficiente* X^{grado}
      - donde coeficiente es un número real y grado un número natural.
    - **Operadores externos de la clase Monomio**

- **PRIMERA PARTE. Clase Monomio**
  - Se codificará la clase en dos ficheros:
    - **Monomio.hpp**: prototipos de las funciones de la clase Monomio
    - **Monomios.cpp**: código de las funciones de la clase Monomio
  - **Observación**:
    - Se debe utilizar una **cota de error** para controlar la precisión de los números reales cuando se comparen.
  - **Atributos**
    - Número real que represente el coeficiente del monomio.
    - Número natural, es decir, número entero mayor o igual que cero, que represente el grado del monomio.
  - **Constructores**
    - **Constructor parametrizado con valores por defecto**[1]
      - ***Monomio (coeficiente: Real=0.0; grado: Entero=0)***
        - Crea un nuevo monomio usando los argumentos
          - *coeficiente* X^{grado}
      - Precondición
        - El grado es mayor o igual que 0.
      - Postcondición
        - El coeficiente del monomio es igual al valor del parámetro *"coeficiente"*.
        - El grado del monomio es igual al valor del parámetro *"grado"*.
    - **Constructor de copia**
      - ***Monomio(m: Monomio)***
        - Crea un nuevo monomio a partir de otro monomio.
      - Postcondición

1Se utiliza notación de pseudocódigo para especificar el prototipo de las funciones.

        - El coeficiente del monomio creado es igual al coeficiente del monomio *"m"*.
        - El grado del monomio creado es igual al grado del monomio *"m"*.

- **Observadores**
  - **Operaciones de consulta de los atributos del monomio**
    - ***Real getCoeficiente()***
      - Obtiene el coeficiente del monomio.
    - ***Entero getGrado()***
      - Obtiene el grado del monomio.
  - **Observación**
    - En C++, estas funciones deben tener el calificador ***const***

- **Modificadores**
  - ***Operaciones* de modificación de los atributos del monomio**
    - ***setCoeficiente(x: Real)***
      - Asigna un nuevo valor *"x"* al coeficiente del monomio.
      - Postcondición:
        - El coeficiente del monomio es igual al número real *"x"*.
    - ***setGrado(n: Entero)***
      - Asigna un nuevo valor *"n"* al grado del monomio.
      - Precondición
        - El número entero *"n"* es mayor o igual que 0.
      - Postcondición
        - El grado del monomio es igual al número entero *"n"*.

  - **Operadores de asignación**
    - ***Monomio operador = (m: Monomio)***
      - Devuelve el monomio actual que ha sido modificado con los atributos del monomio *"m"*.
      - Postcondición
        - El coeficiente del monomio es igual al coeficiente del monomio *"m"*.
        - El grado del monomio es igual al grado del monomio *"m"*.
    - ***Monomio operador = (x: Real)***
      - Devuelve el monomio actual que ha sido modificado para que su grado sea 0 y su coeficiente sea el número real *"x"*.
      - Postcondición
        - El grado del monomio es igual 0.
        - El coeficiente del monomio es igual al número real *"x"*.

  - ***Operadores combinados de operación aritmética y asignación***
    - ***Monomio operador += (m: Monomio)***
      - Modifica el monomio sumándole otro monomio de igual grado.
      - Precondición
        - El monomio *"m"* tiene el mismo grado que el monomio actual.
      - Postcondición
        - El coeficiente del monomio actual se ha incrementado con el coeficiente del monomio *"m"*.

        - El grado del monomio actual no ha sido modificado.
    - ***Monomio operador -= (m: Monomio)***
        - Modifica el monomio restándole otro monomio de igual grado.
        - Precondición
            - El monomio *"m"* tiene el mismo grado que el monomio actual.
        - Postcondición
            - El coeficiente del monomio actual se ha decrementado con el coeficiente del monomio "m".
            - El grado del monomio actual no ha sido modificado.
    - ***Monomio operador *= (m: Monomio)***
        - Modifica el monomio multiplicado por otro monomio.
        - Postcondición
            - El coeficiente del monomio actual se ha multiplicado por el coeficiente del monomio *"m"*.
            - El grado del monomio actual se ha incrementado con el grado del monomio *"m"*.
    - ***Monomio operador /= (m: Monomio)***
        - Modifica el monomio dividiéndolo por otro monomio.
        - Precondición
            - El grado del monomio *"m"* es igual o inferior al grado del monomio actual.
            - El coeficiente del monomios *"m"* no es 0.0.
        - Postcondición
            - El coeficiente del monomio actual se ha decrementado con el coeficiente del monomio *"m"*.
            - El grado del monomio actual se ha incrementado con el grado del monomio *"m"*.
    - ***Monomio operador *= (x: Real)***
        - Modifica el monomio multiplicándolo por un número real.
        - Postcondición
            - El coeficiente del monomio actual se ha multiplicado por el número real *"x"*.
            - El grado del monomio actual no ha sido modificado.
    - ***Monomio operador /= (x: Real)***
        - Modifica el monomio dividiéndolo por un número real.
        - Precondición
            - El número real *"x"* no es 0.0
        - Postcondición
            - El coeficiente del monomio actual se ha dividido por el número real *"x"*.
            - El grado del monomio actual no ha sido modificado.

- **Funciones de lectura y escritura**
    - ***leerMonomio()***
        - Lee desde el teclado los atributos del monomio.
        - Postcondición
            - El grado del monomio es mayor o igual que 0.
    - ***escribirMonomio()***

- Escribe por pantalla los atributos del monomio con el formato:
  - *coeficiente X^ grado*
  - Notas:
    - Si el coeficiente es 1 entonces se escribirá *X^grado*
    - Si el coeficiente es -1 entonces se escribirá *-X^grado*
    - Si el grado es 0 entonces se escribirá solo el *coeficiente*.
    - Si el grado es 1 entonces se escribirá *X* pero sin grado

- **Funciones auxiliares**
  - ***Real calcularValor(x: Real)***
    - Calcula el valor del Monomio para un número real *"x"*:
      - $coeficiente\, x^{grado}$

- **SEGUNDA PARTE. Operadores externos de la clase Monomio**
  - Estos operadores no pertenecen a la clase Monomio pero utilizan objeto de dicha clase.
  - Se codificarán en los ficheros:
    - **operadoresMonomio.hpp**: prototipos de las funciones
    - **operadoresMonomio.cpp**: código de las funciones
  - **Observación**:
    - Se debe utilizar una **cota de error** para controlar la precisión de los números reales cuando se comparen.
  - **Operadores de igualdad**
    - ***Lógico operador == (m1: Monomio; m2: Monomio)***
      - Comprueba si dos monomios son iguales: *m1 == m2*
      - Postcondición
        - El valor devuelto es
          - ✔ verdadero si los dos monomios tienen el mismo grado y el mismo coeficiente;
          - ✔ falso, en caso contrario.
    - ***Lógico operador == (m : Monomio; x : Real)***
      - Comprueba si un monomio es igual a un número real: *m== x*
      - Postcondición
        - El valor devuelto es
          - ✔ verdadero si el monomio tiene grado 0 y su coeficiente es igual al número *"x"*;
          - ✔ falso, en caso contrario.
    - ***Lógico operador == (x: Real; m: Monomio)***
      - Comprueba si un monomio es igual a un número real: *x == m*
      - Postcondición
        - El valor devuelto es
          - ✔ verdadero si el monomio tiene grado 0 y su coeficiente es igual al número *"x"*;
          - ✔ falso, en caso contrario.
  - **Operadores de desigualdad**
    - *Lógico operador != (m1: Monomio; m2: Monomio)*
      - Comprueba si dos monomios no son iguales: *m1 != m2*

- Postcondición
  - El valor devuelto es
    - ✔ verdadero si los dos monomios no tienen el mismo grado o no tienen el mismo coeficiente;
    - ✔ falso, en caso contrario.
- *Lógico operador != (x: Real; m: Monomio)*
  - Comprueba si un monomio no es igual a un número real: *m == x*
  - Postcondición
    - El valor devuelto es
      - ✔ verdadero si el monomio no tiene grado 0 o si su coeficiente no es igual al número *"x"*;
      - ✔ falso, en caso contrario.
- *Lógico operador != (m : Monomio; x: Real)*
  - Comprueba si un monomio no es igual a un número real: *x == m*
  - Postcondición
    - El valor devuelto es
      - ✔ verdadero si el monomio no tiene grado 0 o si su coeficiente no es igual al número *"x"*;
      - ✔ falso, en caso contrario.

- **Operador aritméticos unarios prefijos**
  - *Monomio operador + (m: Monomio)*
    - Devuelve una copia del Monomio *"m"*: *+m*
    - Postcondición
      - El monomio devuelto es igual al monomio "m", es decir, tiene el mismo grado y el mismo coeficiente.
  - *Monomio operador - (m: Monomio)*
    - Devuelve el opuesto del Monomio *"m"*: *-m*
    - Postcondición
      - El monomio devuelto tiene el mismo grado que el monomio *"m"* pero su coeficiente es el opuesto.
- **Operador aritméticos binarios**
  - *Monomio operador + (m1: Monomio; m2: Monomio)*
    - Devuelve otro monomio que es la suma de dos monomios: *m1 + m2*
    - Precondición
      - Los monomios *m1* y *m2* tienen el mismo grado.
    - Postcondición
      - El monomio devuelto tiene
        - ✔ el mismo grado que los monomios *m1* y *m2*,
        - ✔ y su coeficiente es la suma de los coeficientes de los monomios.
  - *Monomio operador - (m1: Monomio; m2: Monomio)*
    - Devuelve otro monomio que es la resta de dos monomios: *m1 – m2*
    - Precondición
      - Los monomios *m1* y *m2* tienen el mismo grado.
    - Postcondición
      - El monomio devuelto tiene el mismo grado que los monomios *m1* y *m2*,
      - y su coeficiente es la resta de los coeficientes de los monomios *m1* y *m2*.
  - *Monomio operador * (m1: Monomio; m2: Monomio)*

- Devuelve otro monomio que es el producto de dos monomios: *m1 * m2*
- Postcondición
  - El monomio devuelto tiene un grado que es la suma de los grados de los monomios *m1* y *m2*,
  - y su coeficiente es el producto de los coeficientes de los monomios *m1* y *m2*.

- *Monomio operador * (m: Monomio; x: Real)*
  - Devuelve otro monomio que es el producto de un monomio por un número real: *m * x*
  - Postcondición
    - El monomio devuelto tiene el mismo grado que el monomio *"m"*,
    - y su coeficiente es el producto del coeficiente del monomio "m" por el número *"x"*.
- *Monomio operador * (x: Real; m: Monomio)*
  - Devuelve otro monomio que es el producto de un monomio por un número real: *x * m*
  - Postcondición
    - El monomio devuelto tiene el mismo grado que el monomio *"m"*,
    - y su coeficiente es el producto del coeficiente del monomio *"m"* por el número *"x"*.
- *Monomio operador / (m1: Monomio; m2: Monomio)*
  - Devuelve otro monomio que es la división de dos monomios: *m1 / m2*
  - Precondición
    - El grado del monomio *m2* es menor o igual que el grado del monomio *m1*.
    - El coeficiente del monomio *m2* no es 0.0.
  - Postcondición
    - El monomio devuelto tiene un grado que es la resta de los grados de los monomios *m1* y *m2*,
    - y su coeficiente es la división de los coeficientes de los monomios *m1* y *m2*.
- *Monomio operador / (m: Monomio; x: Real)*
  - Devuelve otro monomio que es la división del monomio *"m"* por el número real "x": *m / x*
  - Precondición
    - El número *"x"* no es 0.0.
  - Postcondición
    - El monomio devuelto tiene el mismo grado que el monomio *"m"*,
    - y su coeficiente es la división del coeficiente del monomio *"m"* por el número *"x"*.
- *Monomio operador / (x: Real: m: Monomio)*
  - Devuelve otro monomio que es la división del número *"x"* por el monomio *"m"*: *x / m*
  - Precondición
    - El grado del monomio es 0 y su coeficiente es distinto de 0.0.
  - Postcondición
    - El monomio devuelto tiene grado igual a 0

        - y su coeficiente es la división del número "*x*" por el coeficiente del monomio "*m*".
    - **Sobrecarga del operador de flujo de entrada**
        - Lee desde el flujo de entrada los atributos de un monomio separados por espacios
        - Prototipo de C++
            - *istream &**operator**>>(istream &stream, Monomio &v);*
    - **Sobrecarga del operador de flujo de salida**
        - Escribe en el flujo de salida los atributos del monomio separados por espacios
            - Ejemplo: *coeficiente grado*
        - Prototipo de C++
            - *ostream &**operator**<<(ostream &stream, Monomio const &v);*

- **ENTREGA Y EVALUACIÓN**
    - Duración de la práctica nº 1: tres sesiones de dos horas cada una.
    - **Plazo máximo de entrega**
        - **16:00 horas del lunes 4 de marzo de 2019**
    - Se proporciona un fichero comprimido denominado "practica-1-usuario.zip" que contiene los siguientes ficheros
        - **Practica-1.pdf**
            - Enunciado de la práctica 1 (este documento)
        - **makefile**
            - make:
                - Compila el código y crea un programa ejecutable denominado "principal.exe" que permite probar la implementación de la clase Monomio.
            - make ndebug:
                - Compila el código sin incluir los asertos de comprobación de las pre y postcondiciones
            - make doc:
                - Genera la documentación de doxygen
            - make clean:
                - Borra ficheros superfluos
        - **Doxyfile**
            - Fichero de configuración de doxygen
        - **macros.hpp**
            - Permite utilizar macros de pantalla
        - **principal.cpp**
            - Programa de prueba de la práctica 1
        - **funcionesAuxiliares.hpp** y **funcionesAuxiliares.cpp**
            - Prototipo y código de las funciones auxiliares del programa principal
        - **Monomio.hpp** y **Monomio.cpp**
            - Ficheros que permiten implementar la clase Monomio.
            - Estos ficheros deben ser completados por cada estudiante.
        - **operadoresMonomios.hpp** y **operadoresMonomios.cpp**
            - Ficheros que permiten implementar los operadores externos de la clase Monomio.
            - Estos ficheros deben ser completados por cada estudiante.

- **Al terminar la práctica**,
  - se deberá subir un fichero **comprimido** denominado “practica-1-**usuario**.zip”,
  - donde “usuario” es el **login** de cada estudiante.
  - y que contenga todos los ficheros de la práctica.

- **Observaciones**
  - Se debe usar el espacio de nombres de la asignatura: **ed**
  - Se deben utilizar directivas de control para la especificación de los asertos de las pre y post – condiciones.
  - Los prototipos de las funciones se deben comentar con **doxygen**
  - Se debe comentar el código entre líneas.

- **Evaluación**
  - La calificación de la práctica se basará
    - en la calidad y completitud del trabajo realizado.
    - y en la **defensa presencial de cada estudiante**.
  - **Se valorará**
    - La correcta implementación de la clase **Monomio**
    - El correcto funcionamiento del programa principal propuesto como ejemplo.
    - La ampliación y mejora del menú del programa principal para añadir más opciones.
    - La documentación del código con doxygen.
    - La claridad del código.
    - El uso de macros de pantalla para mejorar la visualización de la información
    - **Y sobre todo**
      - Un profundo conocimiento de la práctica codificada.