



Universidad de Córdoba
Escuela Politécnica Superior de Córdoba



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

SEGUNDA PRÁCTICA

POLINOMIOS

- **OBJETIVO**
 - **Codificar en C++**
 - El tipo abstracto de datos **Polinomio**, compuesto por la suma de varios monomios. Un monomio compuesto por coeficiente y grado
 - $\text{coeficiente}X^{\text{grado}}$
 - donde coeficiente es un número real y grado un número natural.
 - **Operadores externos de la clase Polinomio**
 - Utilizar la clase **Monomio** y sus **funciones**, implementadas en la primera práctica.
- **PRIMERA PARTE. Clase Polinomio**
 - Se codificará la clase en dos ficheros:
 - **Polinomio.hpp**: prototipos de las funciones de la clase Polinomio.
 - **Polinomio.cpp**: código de las funciones de la clase Polinomio.
 - **Observación:**
 - Se debe utilizar una **cota de error** para controlar la precisión de los números reales cuando se comparen.
 - **Atributos**
 - Vector de monomios para representar el polinomio. No puede haber dos monomios con el mismo grado. Recordad que cada monomio tenía en su clase los 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**
 - **Polinomio ()**
 - Crea un nuevo monomio usando coeficiente 0.0 y grado 0
 - Postcondición
 - El polinomio creado es nulo.
 - **Constructor de copia**
 - **Monomio(p: Polinomio)**
 - Crea un nuevo polinomio a partir de otro polinomio.
 - Postcondición
 - El polinomio creado es igual al polinomio “p”.
 - **Observadores**
 - **Operaciones de consulta de los atributos del polinomio**

- **Lógico esNulo()**
 - Comprueba si el Polinomio es igual al monomio de coeficiente 0.0 y grado 0.
- **Entero getGrado()**
 - Obtiene el grado del polinomio.
 - Precondición
 - Los monomios están ordenados de mayor a menor grado.
- **Entero getNumeroMonomios()**
 - Obtiene el número de monomios del polinomio.
- **Lógico existeMonomio(n: Entero)**
 - Comprueba si existe el monomio de grado indicado.
 - Precondición
 - El polinomio debe existir.
- **Monomio getMonomio(n: Entero)**
 - Obtiene el monomio de grado indicado.
 - Precondición
 - El polinomio debe existir.
- **Observación**
 - En C++, estas funciones deben tener el calificador *const*
- **Operadores de asignación**
 - **Polinomio operador = (p: Polinomio)**
 - Operador de asignación. Operador que copia un polinomio en el polinomio actual.
 - Precondición
 - El polinomio “p” debe ser diferente del polinomio actual
 - Postcondición
 - El polinomio actual debe ser igual al polinomio “p”.
 - **Polinomio operador = (m: Monomio)**
 - Operador de asignación. Operador que copia un monomio en el polinomio actual.
 - Postcondición
 - El polinomio actual debe tener un único monomio que será igual al monomio “m”.
 - **Polinomio operador = (x: Real)**
 - Operador de asignación. Operador que copia un número en el polinomio actual.
 - Postcondición
 - El polinomio actual debe ser igual al número “x”.
- **Operadores combinados de operación aritmética y asignación**
 - **Polinomio operador += (p: Polinomio)**
 - Operador de incremento y asignación con otro polinomio.
 - **Polinomio operador += (m: Monomio)**
 - Operador de incremento y asignación con un monomio.
 - **Polinomio operador += (x: Real)**
 - Operador de incremento y asignación con un número real.
 - **Polinomio operador -= (p: Polinomio)**

- Operador de decremento y asignación con otro polinomio.
- ***Polinomio operador -= (m: Monomio)***
 - Operador de decremento y asignación con un monomio.
- ***Polinomio operador -= (x: Real)***
 - Operador de decremento y asignación con un número real.
- ***Polinomio operador *= (p: Polinomio)***
 - Operador de multiplicación y asignación por otro polinomio.
- ***Polinomio operador *= (m: Monomio)***
 - Operador de multiplicación y asignación por un monomio.
- ***Polinomio operador *= (x: Real)***
 - Operador de multiplicación y asignación por un número real.
- ***Polinomio operador /= (p: Polinomio)***
 - Operador de división y asignación por otro polinomio.
 - Precondición
 - El grado de polinomio actual es mayor o igual que el grado del polinomio “p”.
 - El polinomio “p” no es nulo.
- ***Polinomio operador /= (m: Monomio)***
 - Operador de división y asignación por un monomio.
 - Precondición
 - El grado del monomio debe ser menor o igual que el grado del polinomio actual.
 - El polinomio actual no es nulo.
- ***Polinomio operador /= (x: Real)***
 - Operador de división y asignación por un número real.
- **Funciones de lectura y escritura**
 - ***leerPolinomio()***
 - Lee desde el teclado un polinomio.
 - ***escribirPolinomio()***
 - Escribe por pantalla el polinomio donde cada monomio tiene el formato:
 - *coeficiente X^ grado*
 - Notas:
 - Si el coeficiente es 1 entonces se escribirá X^{grado}
 - Si el coeficiente es -1 entonces se escribirá $-X^{\text{grado}}$
 - 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 polinomio para un número real “x”

- **SEGUNDA PARTE. Operadores externos de la clase Polinomio**

- Estos operadores no pertenecen a la clase Polinomio pero utilizan objeto de dicha clase.
- Se codificarán en los ficheros:
 - **operadoresExternosPolinomio.hpp**: prototipos de las funciones
 - **operadoresExternosPolinomio.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 == (p1: Polinomio; p2: Polinomio)**
 - Comprueba si dos polinomios son iguales: $p1 == p2$. Si tienen los mismos monomios.
 - Postcondición
 - El valor devuelto es:
 - ✓ verdadero si los dos polinomios tienen los mismos monomios;
 - ✓ falso, en caso contrario.
 - **Lógico operador == (p: Polinomio; m: Monomio)**
 - Comprueba si el polinomio “p” es igual al monomio “m”, es decir, si el polinomio “p” tiene un único monomio que es igual a “m”.
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio tiene un único monomio que es igual a “m”;
 - ✓ falso, en caso contrario.
 - **Lógico operador == (m: Monomio; p: Polinomio)**
 - Comprueba si el polinomio “p” es igual al monomio “m”, es decir, si el polinomio “p” tiene un único monomio que es igual a “m”.
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio tiene un único monomio que es igual a “m”;
 - ✓ falso, en caso contrario.
 - **Lógico operador == (p: Polinomio; x: Real)**
 - Comprueba si el polinomio “p” es igual al número “x”, es decir, si el polinomio “p” tiene un único monomio que es igual a “x”.
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio tiene un único monomio que es igual a “x”;
 - ✓ falso, en caso contrario.
 - **Lógico operador == (x: Real; p: Polinomio)**
 - Comprueba si el polinomio “p” es igual al número “x”, es decir, si el polinomio “p” tiene un único monomio que es igual a “x”.
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio tiene un único monomio que es igual a “x”;
 - ✓ falso, en caso contrario.
- **Operadores de desigualdad**

- **Lógico operador \neq ($p1$: Polinomio; $p2$: Polinomio)**
 - Comprueba si dos polinomios no son iguales: $p1 \neq p2$. Si no tienen los mismos monomios.
 - Postcondición
 - El valor devuelto es:
 - ✓ verdadero si los dos polinomios no tienen los mismos monomios;
 - ✓ falso, en caso contrario.
- **Lógico operador \neq (p : Polinomio; m : Monomio)**
 - Comprueba si el polinomio " p " es no igual al monomio " m ", es decir, si el polinomio " p " no tiene un único monomio que es igual a " m ".
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio no tiene un único monomio que es igual a " m ";
 - ✓ falso, en caso contrario.
- **Lógico operador \neq (m : Monomio; p : Polinomio)**
 - Comprueba si el polinomio " p " no es igual al monomio " m ", es decir, si el polinomio " p " no tiene un único monomio que es igual a " m ".
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio no tiene un único monomio que es igual a " m ";
 - ✓ falso, en caso contrario.
- **Lógico operador \neq (p : Polinomio; x : Real)**
 - Comprueba si el polinomio " p " no es igual al número " x ", es decir, si el polinomio " p " no tiene un único monomio que es igual a " x ".
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio no tiene un único monomio que es igual a " x ";
 - ✓ falso, en caso contrario.
- **Lógico operador \neq (x : Real; p : Polinomio)**
 - Comprueba si el polinomio " p " no es igual al número " x ", es decir, si el polinomio " p " no tiene un único monomio que es igual a " x ".
 - Postcondición
 - El valor devuelto es
 - ✓ verdadero si el polinomio no tiene un único monomio que es igual a " x ";
 - ✓ falso, en caso contrario.
- **Operadores aritméticos unarios prefijos**
 - **Polinomio operador $+$ (p : Polinomio)**
 - Devuelve una copia del Polinomio " p ": $+p$
 - Postcondición
 - El polinomio devuelto es igual al polinomio " p ", es decir, tiene los mismos monomios.
 - **Polinomio operador $-$ (m : Polinomio)**
 - Devuelve el opuesto del Polinomio " p ": $-p$

- Postcondición
 - El polinomio devuelto tiene todos sus monomios opuestos, es decir, el mismo grado pero su coeficiente opuesto.
- **Operadores aritméticos binarios**
 - *Polinomio operador + (p1: Polinomio; p2: Polinomio)*
 - Devuelve otro polinomio que es la suma de dos polinomios: $p1 + p2$
 - *Polinomio operador + (p: Polinomio; m: Monomio)*
 - Devuelve otro polinomio que es la suma de un polinomio y un monomio.
 - *Polinomio operador + (m: Monomio; p: Polinomio)*
 - Devuelve otro polinomio que es la suma de un polinomio y un monomio.
 - *Polinomio operador + (p: Polinomio; x: Real)*
 - Devuelve otro polinomio que es la suma de un polinomio y un número real.
 - *Polinomio operador + (x: Real; p: Polinomio)*
 - Devuelve otro polinomio que es la suma de un polinomio y un número real.
 - *Polinomio operador - (p1: Polinomio; p2: Polinomio)*
 - Devuelve otro polinomio que es la resta de dos polinomios: $p1 - p2$
 - *Polinomio operador - (p: Polinomio; m: Monomio)*
 - Devuelve otro polinomio que es la resta de un polinomio y un monomio.
 - *Polinomio operador - (m: Monomio; p: Polinomio)*
 - Devuelve otro polinomio que es la resta de un polinomio y un polinomio.
 - *Polinomio operador - (p: Polinomio; x: Real)*
 - Devuelve otro polinomio que es la resta de un polinomio y un número real.
 - *Polinomio operador - (x: Real; p: Polinomio)*
 - Devuelve otro polinomio que es la resta de un número real y un polinomio.
 - *Polinomio operador * (p1: Polinomio; p2: Polinomio)*
 - Devuelve otro polinomio que es la multiplicación de dos polinomios: $p1 * p2$
 - *Polinomio operador * (p: Polinomio; m: Monomio)*
 - Devuelve otro polinomio que es la multiplicación de un polinomio y un monomio.
 - *Polinomio operador * (m: Monomio; p: Polinomio)*
 - Devuelve otro polinomio que es la multiplicación de un polinomio y un polinomio.
 - *Polinomio operador * (p: Polinomio; x: Real)*
 - Devuelve otro polinomio que es la multiplicación de un polinomio y un número real.
 - *Polinomio operador * (x: Real; p: Polinomio)*
 - Devuelve otro polinomio que es la multiplicación de un número real y un polinomio.
 - *Polinomio operador / (p1: Polinomio; p2: Polinomio)*
 - Devuelve otro polinomio que es la división de dos polinomios: $p1 / p2$
 - Precondición
 - El grado de $p2$ es menor o igual que el grado de $p1$
 - *Polinomio operador / (p: Polinomio; m: Monomio)*
 - Devuelve otro polinomio que es la división de un polinomio y un monomio.
 - Precondición
 - El grado del monomio es menor o igual que el grado del polinomio

- *Polinomio operador / (m: Monomio; p: Polinomio)*
 - Devuelve otro polinomio que es la división de un polinomio y un polinomio.
 - Precondición
 - El grado del polinomio es menor o igual que el grado del monomio
- *Polinomio operador / (p: Polinomio; x: Real)*
 - Devuelve otro polinomio que es la división de un polinomio y un número real.
 - Precondición
 - El número “x” no es 0.0
- *Polinomio operador / (x: Real; p: Polinomio)*
 - Devuelve otro polinomio que es la división de un número real y un polinomio.
 - Precondición
 - El polinomio tiene grado 0 y su coeficiente no es 0.0
- **Sobrecarga del operador de flujo de entrada**
 - Lee desde el flujo de entrada un polinomio
 - Prototipo de C++
 - *istream &operator>>(istream &stream, Polinomio &p);*
- **Sobrecarga del operador de flujo de salida**
 - Escribe en el flujo de el polinomio
 - Prototipo de C++
 - *ostream &operator<<(ostream &stream, Polinomio const &p);*
- **ENTREGA Y EVALUACIÓN**
 - Duración de la práctica nº 2: tres sesiones de dos horas cada una.
 - **Plazo máximo de entrega**
 - **22:00 horas del domingo 31 de marzo de 2019**
 - Se proporciona un fichero comprimido denominado “practica-2-usuario.zip” que contiene los siguientes ficheros
 - **Practica-2.pdf**
 - Enunciado de la práctica 2 (este documento)
 - **makefile**
 - make:
 - Compila el código y crea un programa ejecutable denominado “principal.exe” que permite probar la implementación de la clase Polinomio.
 - 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 2
- **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.
- **Polinomio.hpp y Polinomio.cpp**
 - Ficheros que permiten implementar la clase Polinomio.
 - Estos ficheros deben ser completados por cada estudiante.
- **operadoresExternosMonomios.hpp y operadoresExternosMonomios.cpp**
 - Ficheros que permiten implementar los operadores externos de la clase Monomio.
 - Estos ficheros deben ser completados por cada estudiante.
- **operadoresExternosPolinomios.hpp y operadoresExternosPolinomios.cpp**
 - Ficheros que permiten implementar los operadores externos de la clase Polinomio.
 - Estos ficheros deben ser completados por cada estudiante.
- **Al terminar la práctica,**
 - se deberá subir un fichero **comprimido** denominado “practica-3-**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 **Polinomio y 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.