ESCUELA DE CIENCIAS DE LA
COMPUTACION E INFORMATICA
UNIVERSIDAD DE COSTA RICA

CI-1201 PROGRAMACION 2
I semestre de 2016
Prof. Alan Calderón Castro

Primer proyecto de programación

Objetivos

Al completar esta tarea el estudiante debería ser capaz de:

1. Manejar con fluidez las carácterísticas relevantes del ambiente de trabajo NetBeans.
2. Comprender y aplicar el esquema de trabajo “Controlador-Modelo”.
3. Construir clases en C++ especificando métodos y atributos.
4. Aplicar la técnica “programación orientada por pruebas” (“test-driven-programming”) que implica usar las pruebas de las clases como especificaciones precisas de las mismas.
5. Usar memoria dinámica efectiva y eficientemente para la construcción de clases en C++.

Descripción del problema

Usted construirá un programa que evalúe una expresión aritmética de polinomios en dos variables (X,Y), sin paréntesis y con los operadores usuales (+, -, *, ...). Algunas expresiones aritméticas de ejemplo son:

1) expresiones con un único operador: p1*p2, p1+p2,p1-p2,p1/p2
2) expresiones con dos operadores de igual prioridad: p1+p2-p3, p1-p3+p2, p1*p2/p3
3) expresiones con operadores de diferente prioridad: p1*p2+p3, p1-p2*p3, p1/p2-p3

Las variables “p1”, “p2” y “p3” representan todas polinomios en dos variables (X,Y) de grado N >= 0. Su programa deberá respetar las reglas de prioridad de los operadores. Un archivo de datos de texto simple (*.txt creado con NotePad) contendrá al principio líneas que asocian un nombre a un polinomio, luego una lista de expresiones a evaluar todas basadas en los nombres de polinomios de la primera lista. Su programa deberá analizar y procesar cada línea del archivo de datos. Se adjunta un archivo de datos.

Su programa tendrá un archivo con la función “main()”, una clase “Polinomio”, una clase “Evaluador”, una clase “Mapeo” y un programa de pruebas (tests) para cada una de las clases Polinomio y Evaluador. Se proveen los archivos de firmas correspondientes. Las siguientes son las funciones de cada componente:

1. “main()” deberá:
1.1. leer el archivo de datos,
1.2. cada línea de definición de un polinomio deberá pasarla a un constructor adecuado de la clase “Polinomio”,
1.3. cada línea de expresión de polinomios deberá pasarla a la única instancia de la clase “Evaluador” usando algún método adecuado,
1.4. el resultado de la evaluación de una expresión será recibido por el “main()” como una instancia de “Polinomio”, la transformará en una “string” que luego imprimirá sobre un archivo de texto simple de salida,
1.5. para lograr #1.3 deberá crear una instancia de la clase “Evaluador”, además de una instancia de la clase “Mapeo” para guardar los nombres y sus correspondientes polinomios asociados.

2. Clase “Polinomio” deberá:
2.1 crear instancias de “Polinomio” por omisión como el polinomio cero: 0,
2.2 crear copias de instancias de “Polinomio”,
2.3 crear instancias de “Polinomio” a partir del análisis de “strings” como las primeras cuatro líneas del archivo de datos de ejemplo,
2.4 representar simplificadamente y en orden descendente por grado de X y Y, cada instancia de polinomio que construya ya sea a través de los constructores (estándar, de copias y el que recibe una “string” como parámetro), así como las instancias creadas por medio de los operadores,
2.5 realizar las operaciones básicas sobre polinomios (suma, resta, multiplicación y división),
2.6 transformar cualquier instancia simplificada de clase en una “string” legible para las personas.

3. Clase “Monomio”: se usará la provista en el código base.

4. Clase “Mapeo”: su función es simplemente guardar las asociaciones entre los nombres y las instancias de Polinomio que sean necesarias según el archivo de datos para entregarlas (mediante el método obtPolinomio(...)) a la instancia de “Evaluador” en el momento en que sea necesario para la evaluacion de expresiones.

5. Clase “Evaluador” deberá:
5.1 Procesar correctamente cada expresión de polinomios como por ejemplo las del segundo grupo del archivo de datos de ejemplo,
5.2 Cada vez que reciba una expresión de polinomios devolverá una nueva instancia de la clase “Polinomio” con el resultado, para esto se basará en los operadores de la clase “Polinomio”.
5.3 Se basará en el algoritmo de evaluación de expresiones discutido en clase.

Procedimiento de resolución

Este es un laboratorio guiado, por tanto se deberán seguir los siguientes pasos en el orden aquí estipulado, lo cual será continuamente verificado por el docente durante las sesiones de clase en el laboratorio.

1. Programar las pruebas de la clase Polinomio: ver archivo adjunto con pruebas mínimas sobre la clase Polinomio.

2. Programar la clase Polinomio
2.1 Programar los constructores.
2.2 Programar el destructor si fuera necesario.
2.3 Programar las funciones miembro observadoras: “verGrado()”, “eval(x,y)”, “aHil()”, “verInv()”.
2.4 Programar los cuatro operadores: +,-,* y /.

NOTA: puede intercalar la aplicación de las pruebas con la programación de cada función miembro o método de la clase.

3. Aplicar las pruebas a la clase Polinomio y depurarla.

4. Programar la clase Mapeo.

5. Programar las pruebas de la clase Evaluador.
5.1 Utilizar el archivo de datos de prueba para comprobar si el método “aPosfija(...)” funciona correctamente.
5.2 Las pruebas sobre el método “evaluarPosfija” se realizarán junto con las del programa principal.

6. Programar la clase Evaluador.

7. Aplicar las pruebas a la clase Evaluador (las del método “aPosfija(...)”) y depurarla.

8. Programar el “main()”.

9. Aplicar los datos de prueba y depurar todo el programa: ver archivo adjunto con datos mínimos para realizar pruebas sobre el programa completo.

Criterios de evaluación

La nota final de su trabajo dependerá de si en su programa se:

1. Respetan las reglas de estilo del código: márgenes, nombres de objetos empiezan en minúsculas, nombres de clases empiezan en mayúsculas, nombres de métodos también empiezan en minúscula pero se usan mayúsculas para concatenar palabras, comentarios para los atributos y variables de métodos.
2. Se documentan mediante REQ, MOD y EFE los principales métodos (omita constructores, destructor, getters y setters).
3. Respeta la división de responsabilidades entre el programa la función “main()” y las demás clases descritas anteriormente (denominadas en conjunto “el modelo”).
4. Ha simplificado el código lo más posible.
5. Usan algoritmos y estructuras de datos eficientes de acuerdo con las pautas determinadas en clase.
6. Incluye pruebas exhaustivas para las clases “Polinomio”, “Monomio” y “Evaluador”.

Nota: se adjunta el código correspondiente.

Productos

A través del sitio del curso, usted deberá entregar:

- Una carpeta comprimida (usando 7z) con el código fuente del proyecto (sólo archivos *.h y *.cpp, tanto del código principal como de los archivos de prueba). Además incluirá un archivo de datos que usó para el programa principal.
- El nombre del archivo **SÓLO** contendrá los números de carnet de los participantes, por ejemplo: “**A12345_A67890.7z**”.
- En caso de que su programa no realice correctamente todas las operaciones descritas, haga un reporte de errores. Para cada error explique: 1) en qué consiste el error, 2) cuál cree usted que es la causa, 3) qué hizo para corregirlo, aunque no haya funcionado. En la medida en que este reporte muestre su dedicación al trabajo su nota final para esta tarea podría mejorar.

Fecha de entrega: **lunes 2 de mayo a las 9:00 horas**.

Evaluación:

1. Respeto de las reglas de estilo y simplificación del código:.....5%
2. División de responsabilidades entre “main()” y “el modelo”:…….…..5%
3. Comprobación exhaustiva, efectividad y eficiencia.....90%
4. Si su programa tiene errores, por cada error correctamente identificado y suficientemente descrito obtendrá hasta 5/100 puntos adicionales sobre la nota.

Notas:
1. Este proyecto deberá realizarse idealmente y a lo más en parejas. NO SE ACEPTARÁ NINGÚN TRABAJO ELABORADO POR MÁS DE DOS PERSONAS.
2. A TODOS LOS ESTUDIANTES INVOLUCRADOS EN UN FRAUDE SE LES APLICARÁ EL ARTÍCULO #5 INCISO C DEL “Reglamento de Orden y Disciplina de los Estudiantes de la Universidad de Costa Rica”.