Reporte de laboratorio 3

Laura Rincón Riveros - B55863
Esteban Vargas Vargas - B16998
Grupo 3

30 de septiembre de 2016

Índice

---

1. Introducción

En el presente laboratorio se utilizó una clase plantilla y la sobrecarga de operadores. Como se presenta a continuación:

2. Desarrollo

2.1. Clase Calculadora

Se creó una clase emplantillada llamada Calculadora, en la cuál se definieron las funciones básicas suma, resta, multiplicación, división e impresión. Esta clase se creó emplantillada para que pudiera realizar operaciones sobre distintos tipos de datos, no sólo tipos primitivos como enteros y decimales sino que también objetos creados como una fracción, una matriz o un polinomio. Esta se presenta a continuación:

```
#ifndef CALCULADORA_H
#define CALCULADORA_H
#include <iostream>
using namespace std;

template <typename T>
class Calculadora{
public:
   ///Constructor vacío de Calculadora
   Calculadora(){
   }
   ///Destructor de Calculadora
   virtual ~Calculadora(){
   }
   ///Función suma de calculadora
   T add(T a, const T b){
      return a+b;
   }
   ///Función resta de calculadora
   T sub(T a, const T b){
      return a-b;
   }
   ///Función multiplicación de calculadora
   T mul(T a, const T b){
      return a*b;
   }
   ///Función división de calculadora
   T div(T a, const T b){
      return a/b;
   }
   ///Función imprimir de calculadora
   T print(const T a){
        cout << a << endl;
   }
};
#endif /* CALCULADORA_H */
```

Figura 1: Calculadora.h

2.2. Clase Fracción

La clase Fracción cuenta con los atributos:

- Numerador (int)
- Denominador (int)

A continuación se muestra la declaración de los atributos y funciones de la clase Fracción, del archivo encabezado de Fracción.

```
#ifndef FRACCION_H
#define FRACCION_H

using namespace std;
#include <iostream>

class Fraccion{
public:

    ///Constructor vacío de Fraccion
    Fraccion();

    ///Constructor de Fraccion
  Fraccion(int numerador, int denominador);

  ///Destructor de Fraccion
  virtual ~Fraccion();

  ///Atributos de Fraccion
  int numerador;
  int denominador;

  ///Funciones de Fraccion
  Fraccion* operator+(const Fraccion&);
  Fraccion* operator-(const Fraccion&);
  Fraccion* operator*(const Fraccion&);
  Fraccion* operator/(const Fraccion&);
  void operator~();

};

#endif /* FRACCION_H */
```

Figura 2: Fracción.h

La implementación de la sobrecarga de operadores para este tipo de objeto se realizó de la siguiente manera:

- Suma: Se tomaron en cuenta 3 casos (Teniendo F1 y F2):

1. Si los denominadores de F1 y F2 son iguales el resultado es una fracción con el denominador y el numerador igual a la suma del numerador de F1 y F2.
2. Si los denominadores de F1 y F2 son distintos, el de F1 es mayor que el de F2 y el de F1 es divisible entre el de F2 entonces el resultado es una fracción con el denominador igual al denominador de F1 y el numerador es igual a la suma del numerador de F1 mas el numerador de F1 multiplicado por la división del denominador de F1 y F2.
3. Si los denominadores D1 y D2 de F1 y F2 son distintos pero no son divisibles entre ellos, entonces el resultado es una fracción con un denominador DS igual a la multiplicación de D1 y D2. El numerador del resultado se obtiene sumando la multiplicación de la división de DS entre D1 y D2 multiplicado por su respectivo numerador.

- Resta: Se implementó de igual manera que la suma pero realizando una resta.
- Multiplicación: Se obtuvo multiplicando los numeradores y denominadores de las dos fracciones a multiplicar.
- División: Sea una fraccion F1 con denominador D1 y numerador N1, y sea una fraccion F2 con denominador D2 y numerador N2, el resultado se compone de un numerador N=N1*D2 y un numerador D=D1*N2.
- Impresión: Se imprime el numerador y el denominador, separados por un /".

2.3. Clase Matriz

La clase Matriz cuenta con los atributos:

- Número de filas (int)
- Número de columnas (int)
- Contenido (int**)

A continuación se muestra la declaración de los atributos y funciones de la clase Matriz, del archivo encabezado de Matriz.

```
#ifndef MATRIZ_H
#define MATRIZ_H

#include <iostream>

class Matriz{
public:

    ///Constructor vacío de Matriz
    Matriz();

    ///Constructor de Matriz
    Matriz(int filas, int columnas, int** contenido);

    ///Destructor de Matriz
    virtual ~Matriz();

    ///Atributos de Matriz
    int filas;
    int columnas;
    int** contenido;

    ///Funciones de Matriz
    Matriz* operator+(const Matriz&);
    Matriz* operator-(const Matriz&);
    Matriz* operator*(const Matriz&);
    Matriz* operator/(const Matriz&);
    void operator~();

};

#endif /* MATRIZ_H */
```

Figura 3: Matriz.h

La implementación de la sobrecarga de operadores para este tipo de objeto se realizó de la siguiente manera:

- Suma: Luego de comprobar que las dimensiones de las dos matrices sean las mismas, se suma entrada por entrada.
- Resta: Luego de comprobar que las dimensiones de las dos matrices sean las mismas, se resta entrada por entrada.
- Multiplicación: Luego de comprobar que el número de columnas de la primera matriz sea igual al número de filas de la segunda matriz, se crea una matriz nueva del número de filas de la primera matriz y del número de columnas de la segunda. Posteriormente se procede a llenar esta nueva matriz resultado con el algoritmo conocido de multiplicación de matrices.
- División: Luego de comprobar que las dimensiones de las dos matrices sean las mismas, se divide entrada por entrada.
- Impresión: Mediante el uso de 2 *for* se recorre toda la matriz entrada por entrada y se despliega en consola.

2.4. Clase Polinomio

La clase polinomio cuenta con los atributos:

- Grado Mayor (int)
- Coeficientes (int*)

A continuación se muestra la declaración de los atributos y funciones de la clase Polinomio, del archivo encabezado de Polinomio.

```
#ifndef POLINOMIO_H
#define POLINOMIO_H

using namespace std;
#include <iostream>

class Polinomio{
public:

    ///Constructor vacío de Polinomio
    Polinomio();

    ///Constructor  de Polinomio
    Polinomio(int gradoMayor, int* coeficientes);

    ///Destructor de Polinomio
    virtual ~Polinomio();

    ///Atributos de Polinomio
    int gradoMayor;
    int* coeficientes;

    ///Funciones de Polinomio
    Polinomio* operator+(const Polinomio&);
    Polinomio* operator-(const Polinomio&);
    Polinomio* operator*(const Polinomio&);
    Polinomio* operator/(const Polinomio&);
    void operator~();
};

#endif /* POLINOMIO_H */
```

Figura 4: Polinomio.h

La implementación de la sobrecarga de operadores para este tipo de objeto se realizó de la siguiente manera:

- Suma: Se consideraron dos casos para verificar cuál de los dos polinomios tenía mayor grado. A partir de ahí, se creó un objeto nuevo con el tamaño del puntero de coeficiente adecuado; se

sumaban los coeficientes de ambos polinomios desde el grado 0 hasta el grado del polinomio menor y luego se agregaban los coeficientes faltantes del polinomio mayor.

- Resta: Se siguió el procedimiento de suma pero restando los coeficientes de los polinomios, esta vez.
- Multiplicación: Para la multiplicación se consideró que el polinomio resultante sería de un grado equivalente a la suma de los grados de los polinomios que la función recibe. Con lo anterior, se procedió a llenar el nuevo polinomio resultante, con la multiplicación de cada término y la suma de los téminos del mismo grado.
- División: Para la división se emplentó un algoritmo de los siguientes pasos:

  Se divide el primer término del dividendo (this− >coeficientes) entre el primer término del divisor(other.coeficientes), para obtener el primer término del cociente (Result.coeficientes). Se Multiplica el divisor por el primer término del cociente y se resta al dividendo el resultado anterior para conseguir el primer residuo parcial (Temp.coeficientes). Y se repite el procedimiento haciendo ahora dividendo el primer resto parcial.La división finaliza cuando el grado del resto es menor que el grado del divisor.
- Impresión: Para el caso de la impresión,se recorrió el arreglo de coeficientes, se imprimía el valor y la posición del arreglo como el grado del polinomio adicionando un + entre cada término.

2.5. Main

En el archivo main se crearon pares de los objetos mencionados Matriz, Polinomio y Fracción para aplicarle las funciones de suma, resta, multiplicación y división; posibles gracias a la sobrecarga de operadores. Para aplicarle las funciones a los objetos se hizo uso de la clase emplantillada calculadora, la cual permite introducirle cualquier tipo de dato. A continuación se ilustra lo anterior.

2.6. Diagrama de clases

Para una mejor visualización de la composición de las clases se presentan a continuación los diagramas respectivos.

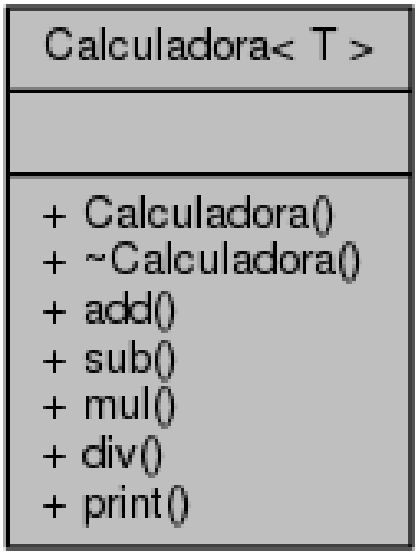

Figura 5: UML Calculadora

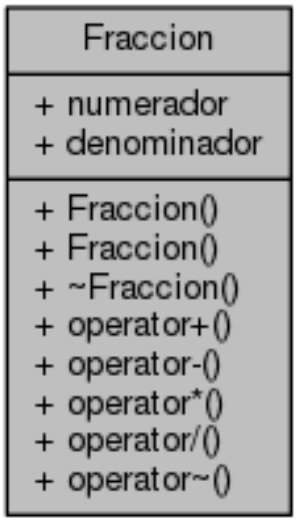

Figura 6: UML Fracción

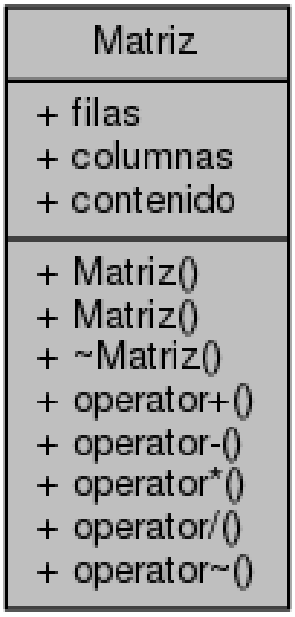

Figura 7: UML Matriz

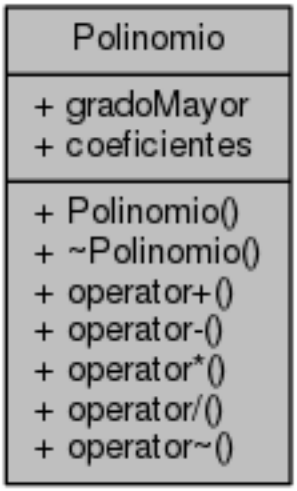

Figura 8: UML Polinomio

3. Conclusiones

- Se empleó la programación orientada a objetos para el uso de clases.
- Se construyó una clase emplantillada capaz de manipular distintos tipos de datos (primitivos y objetos).
- Se sobrecargaron distintos operadores para poder efectuar operaciones básicas a matrices, polinomios y fracciones.