Reporte Tarea 2

Ericka Zúñiga Calvo - B47835

23 de enero de 2017

Índice

---

1. Enunciados

1. Implemente una clase en C++ que modele polinomios, utilizando sobrecarga de operadores que efectúe las cuatro operaciones básicas algebraicas:

   - P& operator+(const P& other);
   - P& operator-(const P& other);
   - P& operator*(const P& other);
   - P& operator/(const P& other);
     Para la división utilice el método de la división sintétiica con sus restricciones.

2. Haga un programa de prueba en donde se creen dos objetos e interactuen entre ellos.

3. Escriba también un Makefile con al menos tres reglas:

   - build: para compilar el programa.
   - run: para ejecutar el programa.
   - clean: para eliminar ejecutables y archivos intermedios.

2. Código

1. Polinomio.h

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

class Polinomio {
```

```
public:

    template<size_t X> Polinomio(int (&arr)[X])
    {
        grado = X-1;
        data = new int[grado+1];
        for (int i=0;i<X;i++)
        {
            data[i]=arr[i];
        }
    }
    Polinomio(int arr[], int sizeArr);
    Polinomio(const Polinomio& orig);
    virtual ~Polinomio();
    void result();

    Polinomio& operator+(const Polinomio &other);
    Polinomio& operator-(const Polinomio &other);
    Polinomio& operator*(const Polinomio &other);
    Polinomio& operator/(const Polinomio &other);

    int grado;
    int* data;

};

#endif /* POLINOMIO_H */
```

2. Polinomio.cpp

   - Se incluye el header de la clase polinomio.

     ```
     #include "Polinomio.h"
     ```

   - Polinomio(int arr[ ],int sizeArr)
     Este es el constructor de la clase Polinomio, recibe como parámetros un arreglo y el tamaño de ese arreglo. Se inicializan los atributos del objeto polinomio.

     ```
     Polinomio::Polinomio(int arr[], int sizeArr) {
         grado = sizeArr-1;
         data = new int[grado+1];
         for (int i=0;i<sizeArr;i++)
         {
             data[i]=arr[i];
         }
     }
     ```

   - Polinomio(const Polinomio &orig)
     Este es el constructor por copia de la clase Polinomio.
     Recibe como parámetro un objeto de la clase Polinomio.

     ```
     Polinomio::Polinomio(const Polinomio& orig) {
         grado = orig.grado;
         data = new int[grado+1];
         for(int i=0;i<orig.grado+1;i++){
             data[i]=orig.data[i];
         }
     }
     ```

- Polinomio()
  Destructor de la clase polinomio. Elimina el arreglo data creado dinamicamente.

```
Polinomio::~Polinomio() {
    delete[] data;
}
```

- result()
  Se encarga de imprimir el polinomio resultante.

```
void Polinomio::result(){
    for(int i=grado;i>0;i--){
        if (data[i-1] >= 0)
        {
            cout<<data[i]<<"x^"<<i<<"+";
        }
        else
        {
            cout<<data[i]<<"x^"<<i;
        }
    }
    cout<<data[0]<<endl;
}
```

- operator +:
  Esta es la sobrecarga del operador +, para poder ser utilizado con objetos de tipo polinomio.
  Inicialmente se compara si el grado de ambos polinomios es igual. Si lo es, simplemente se suman los elementos de la misma posicion en data de cada uno de los polinomios. Y esto es guardado en un arreglo temporal. Dicho arreglo sirve como parámetro para la creación del polinomio resultante.

  Si el grado es diferente, se crea un puntero al polinomio de mayor grado, y se crea una variable menor grado perteneciente al polinomio restante. La suma se realiza como fue especificado anteriormente, hasta que alcance el valor de menor grado. Luego desde menor grado hasta el grado de mayor, se va guardando en la posicion correspondiente el valor de data en esa posicion del puntero mayor.

```
Polinomio& Polinomio::operator+(const Polinomio &other)
{
    if (grado == other.grado)
    {
        int* tempArray = new int[grado+1];

        for(int i=0; i<grado+1;i++)
        {
            tempArray[i]= data[i]+other.data[i];
        }

        Polinomio* result = new Polinomio(tempArray,grado+1);

        delete[] tempArray;
        return *result;
    }
```

```
    else
    {
        int menorGrado;
        const Polinomio* mayor;
        if (other.grado > grado)
        {
            menorGrado = grado;
            mayor = &other;
        }
        else
        {
            menorGrado = other.grado;
            mayor = this;
        }
        int* tempArray = new int[mayor->grado+1];
            for(int i=0; i<menorGrado+1;i++)
            {
                tempArray[i]= data[i]+other.data[i];
            }
            for(int i=menorGrado+1;i<mayor->grado+1;i++)
            {
                tempArray[i]=mayor->data[i];
            }

            Polinomio* result = new Polinomio(tempArray,mayor->grado+1);

            delete[] tempArray;
            return *result;
    }
}
```

- operator -:
  El operador - tiene una gran similitud con el operador +. Pero tiene una condición, si grado mayor es igual a other, entonces en el caso de que los grados de los polinomios sean diferentes, se multiplica por -1, los valores que se encuentran en las posiciones entre menor grado y el grado de mayor.

```
Polinomio& Polinomio::operator-(const Polinomio &other)
{
    if (grado == other.grado)
    {
        int* tempArray = new int[grado+1];

        for(int i=0; i<grado+1;i++)
        {
            tempArray[i]= data[i]-other.data[i];
        }

        Polinomio* result = new Polinomio(tempArray,grado+1);

        delete[] tempArray;
        return *result;
    }
    else
    {
```

```
            int menorGrado;
            const Polinomio* mayor;
            if (other.grado > grado)
            {
                menorGrado = grado;
                mayor = &other;
            }
            else
            {
                menorGrado = other.grado;
                mayor = this;
            }
            int* tempArray = new int[mayor->grado+1];
                for(int i=0; i<menorGrado+1;i++)
                {
                    tempArray[i]= data[i]-other.data[i];
                }
                for(int i=menorGrado+1;i<mayor->grado+1;i++)
                {
                    if(mayor->grado==grado)
                    {
                        tempArray[i]=mayor->data[i];
                    }
                    else
                    {
                        tempArray[i]=mayor->data[i]*(-1);
                    }
                }

                Polinomio* result = new Polinomio(tempArray,mayor->grado+1);

                delete[] tempArray;
                return *result;
        }
    }
```

- operator *:
  Este es el operador * de la función polinomio. Para su solución, inicialmente se crea un arreglo dinamico temporal de tamaño de la suma de los grados de los dos polinomios +1.Posteriormente se van multiplicando cada uno de los elementos del data del polinomio1, con cada uno de los elementos del data del polinomio 2. Y estos se van guardando (+=) en la posicion en el arreglo temporal que corresponde a la suma de cada una de las posiciones en el data de sus polinomios. Se crea un objeto de tipo polinomio, que utiliza como atributo el arreglo y su tamaño.

```
Polinomio& Polinomio::operator*(const Polinomio &other)
{
    int nuevoGrado = grado+other.grado;
    int* tempArray = new int[nuevoGrado+1] ();

        for(int i=0; i<grado+1;i++)
        {
            for (int j=0;j<other.grado+1;j++)
            {
                int temp = i+j;
                tempArray[temp]+= data[i]*other.data[j];
            }
```

```
        }

        Polinomio* result = new Polinomio(tempArray,nuevoGrado+1);

        delete[] tempArray;
        return *result;

}
```

- operator /:
  Este es el operador / de la clase polinomio. Este es construido con el método de división sintética, el cual cuenta con las restricciones: El dividendo debe tener grado mayor que uno, El divisor debe tener grado igual a 1, y coeficiente en dicha posición de 1.

  Al confirmar dichas restricciones, se inicia con el algoritmo. Se crea un arreglo temporal con tamaño de la resta entre el grado del dividendo y el grado del divisor. Y se toma el coeficiente en la posicion más alta del dividendo, como una variable temporal y esta se guarda en la posición más alta del arreglo temporal. Se inicia un ciclo, que inicia en el grado del dividendo -1 y este va decreciendo hasta alcanzar el valor de 0.Luego la variable temporal se multiplica por el valor en la posición más pequeña del divisor, y se guarda en temp2. Y se realiza la suma entre el elemento en data en la posicion [i] y el valor temp2, esto se guarda en la posicion en el arreglo de i-1 y en la variable temporal. Se retorna un objeto Polinomio que recibe como parámetro el arreglo temporal.

```
Polinomio& Polinomio::operator/(const Polinomio &other){
    Polinomio* result = NULL;
    if(other.grado!=1||grado<1||other.data[1]!=1)
    {
        cout<<"No se puede realizar dicha division por el metodo de division sintetica.'
        result = new Polinomio(NULL,0);
    }
    else{
        int* tempArray = new int[grado-other.grado+1];
        int temp = data[grado];
        int div = other.data[0]*(-1);
        tempArray[grado-other.grado]=temp;
        for(int i=grado-1;i>0;i--){
            int temp2= temp*div;
            tempArray[i-1]=data[i]+temp2;
            temp=tempArray[i-1];

        }
        result = new Polinomio(tempArray,grado-other.grado+1);
        delete[] tempArray;
    }
    return *result;
}
```

3. Main
   Programa de prueba, donde se crean dos objetos de la clase polinomio. Y se suman, restan, multiplican y dividen entre si. Así como se logra imprimir su resultado.

```
#include <cstdlib>
#include "Polinomio.h"
using namespace std;

int main(int argc, char** argv) {
```

```
    int array1[] = {-16,2,0,-8,1};
    int array2[] = {-8,1};

    Polinomio* p1=new Polinomio(array1);
    Polinomio* p2=new Polinomio(array2);

    Polinomio p3 = *p2+*p1;
    p3.result();

    Polinomio p4 = *p2-*p1;
    p4.result();

    Polinomio p5 = (*p1)*(*p2);
    p5.result();

    Polinomio p6 = (*p1)/(*p2);
    p6.result();
    delete p1;
    delete p2;
    return 0;
}
```

3. Conclusiones

- Se logró sobrecargar los operadores suma, resta, multiplicación y división con éxito para la clase Polinomio.