Tarea 2
Operaciones de polinomios

Robin González - B43011

26 de enero de 2017

Índice

---

1. Introducción

En este reporte se describe la implementación del código planteado en la guía de la tarea para realizar operaciones con objetos del tipo Polinomio.

Se mostrará el código para las funcionres redefinidas y el del archivo .hpp.

Todos los archivos se encuentran en el mismo directorio.

2. Clase Polinomio

2.1. Header

```
#include <iostream>

using namespace std;
```

```
class Polinomio
{
   //privado
   double* datos;

   public:
      //constructor
      Polinomio();
      Polinomio(int grado, double coeficientes[] = nullptr);

      //atributos
      int largo;
      double* coefi;

      //metodos
      void print();
      Polinomio& operator+( Polinomio &p);
      Polinomio& operator-( Polinomio &p);
      Polinomio& operator*( Polinomio &p);
      Polinomio& operator/( Polinomio &p);
      Polinomio& operator=( Polinomio &p);

      virtual ~Polinomio();

};
```

2.2. Operadores

A continuación se muestra el código para: sumar, restar, multiplicar y dividir polinomios.

2.2.1. Suma

```
Polinomio& Polinomio::operator+(Polinomio &p){//ampersand es una referencia al objeto no
    el de direccion
   int l1 = largo;
   int l2 = p.largo;
   double *ind;//(l1>l2 ? l1:l2)
   if(largo < l2){
      ind = p.coefi;
      for(int i = 0; i < l1; i++){
         ind[i] = coefi[i] +p.coefi[i];
      }
   }
   else{
      ind = coefi;
      for(int i = 0; i < l2; i++)
      {
         ind[i] = coefi[i] +p.coefi[i];
      }
      l2 = l1;
   }
   Polinomio* polonio = new Polinomio(l2,ind);
```

```
   return *polonio;
}
```

2.2.2. Resta

```
Polinomio& Polinomio::operator-(Polinomio &p){//se le resta la objeto de la izquierda
    con que se llama la fucion
   Polinomio* papalote;
   for(int i = 0;i < p.largo; i++){
      p.coefi[i] = -p.coefi[i];
   }
   papalote = &(*this+p);
   return *papalote;
}
```

2.2.3. Multiplicacion

```
Polinomio& Polinomio::operator*(Polinomio &p){
   double arr[largo + p.largo -1];//largo + p.largo -1 ya que el largo maximo resultante
       es la suma de los anteriores -1
   for(int c = 0; c < largo; c++){
      for(int b = 0; b < p.largo; b++){
         arr[c+b] = coefi[c]*p.coefi[b] + arr[c+b];//iteramos multiplicaciones de
             coeficientes por cada casilla de ambos vectores
      }
   }
   Polinomio* peluca = new Polinomio(largo +p.largo-1,arr);
return *peluca;
}
```

2.2.4. Division

```
Polinomio& Polinomio::operator/(Polinomio &p){
   double cucumber[largo-p.largo];
   Polinomio* cociente = new Polinomio(largo-p.largo,cucumber);
   Polinomio* residuo;
   int c = largo;
   int div = p.largo;
   while(c>=p.largo)
   {
cout << "corriendo" << endl;
cout << "coefi " << coefi[c-1] << endl;
cout << "p.coefi " << p.coefi[div-1] << endl;
cout << "coefi/p.coefi " << coefi[c-1]/p.coefi[div-1] << endl;
      cociente->coefi[c-div -1] = coefi[c-1]/p.coefi[div-1];
cout << cociente->coefi[c-div-1] << endl;
      residuo = &((*cociente)*(p));
cout << "cociente" << endl;
   cociente->print();
```

```
cout << "residuo" << endl;
   residuo->print();
      (*this) = (*this)-(*residuo);
      --c;
   }
cout << "corriendo1" << endl;
   return *cociente;
}
```

2.2.5. Asignacion

```
Polinomio& Polinomio::operator=(Polinomio &p){
   Polinomio* poliatomico = new Polinomio(p.largo,p.coefi);
   largo = p.largo;
   coefi = p.coefi;
   return *poliatomico;
return p;
}
```

2.3. Funciones adicionales

2.4. Print

Una bella funcion print para mostrar los polinomios perfectamente bien.

```
void Polinomio::print(){
   int l = largo;
   for(int e = l-1; e >= 0;e--)
   {
      if(datos[e] != 0)
      {
      if(datos[e] > 0 && e != l-1)cout << "+ ";
         if(e == 0)
         {
            cout << datos[e];
         }else
         {
            cout<< datos[e] << "x^" << e;
         }
      }
      cout << " ";
   }
   cout << endl;
}
```

3. Makefile

Se crearon los comandos solicitados por el profesor:

- make : compila.

- run : corre el ejecutable.
- clean : borra ejecutable y archivos intermedios.

```
CC=gcc
CXX=g++
RM=rm -f
CFLAGS = -Wall -g -lm
LFLAGS =
LIBS =
INCLUDES = -I*.hpp

SRCS= polinomios.cpp main.cpp
OBJS=$(subst .cc,.o,$(SRCS))

MAIN = operaciones

all:   $(MAIN)
   @echo "Felicidades ya compilo :)"

$(MAIN): $(OBJS)
   $(CXX) $(CFLAGS) $(INCLUDES) -o $(MAIN) $(OBJS) $(LFLAGS) $(LIBS)

run:
   @echo ">>>Ejecutando..."
   ./operaciones

clean:
   @echo ">>>Limpiando..."
   $(RM) *.o *~ operaciones
```

4. Conclusiones

Si hubiera empezado antes hubiera terminado antes.