Lab 8

COMP9021, Session 1, 2014

The aim of this lab is to practice the use of functions that read data from a file, the use of structures, the use of complex numbers, and introduce the Portable Pixel Map format.

1 Representation of and operations on polynomials

You will write a program that reads two polynomials stored in a file, say `polys.txt`, the first polynomial being stored on the first line of the file, and the second polynomial being stored on the second line of the file. The name of the file will be provided as a command line argument and the file will be opened for reading using `fopen()`. A monomial is one of:

- a constant;
- `x`;
- an expression of the form *`coefficient`* `x` (with no inserted space);
- an expression of the form `x^`*`exponent`* (with no inserted space);
- an expression of the form *`coefficient`* `x^`*`exponent`* (with no inserted space).

Constants and coefficients are integer constants, represented by `0`, `+0`, `-0`, or by a string of digits that does not start with `0` and that may be directly preceded with either `+` or `-`. Exponents are nonnegative integer constants, represented by `0` or by a string of digits that does not start with `0`.

A polynomial is either a monomial, or sums and differences of monomials. More precisely, a polynomial that is not a monomial is of the form

`monomial_1 op_1 ... monomial_2 op_2 ... monomial_n+1`

where *`op_1`*, ..., *`op_n`* are either `+` or `-`, and where any number of space characters (possible none) can be inserted before *`monomial_1`*, after *`monomial_n+1`*, or between operators and monomials.

For instance, the file `polys.txt` could contain:

```
      1 - x  +  8x^2 -   -1x^4  +  x
x^0+2-6x^4+1x^12 +x^14+7x^4+5-6
```

Monomials and polynomials will be represented as the following structures, with the second member of `Polynomial` expected to be allocated enough memory to store `nb_of_monomials` `Monomials`.

```
typedef struct {
    int coeff;
    int degree;
} Monomial;

typedef struct {
    int nb_of_monomials;
    Monomial *monomials;
} Polynomial;
```

Your program should:

- output the normal form of the polynomials stored in `polys.txt`: a normal form is either `0` or sums and differences of monomials where no two monomials have the same degree, monomials are ordered from larger degrees to lower degrees, coefficients are positive, and coefficients as well as degrees of 1 are left implicit;
- output the polynomial, in normal form, that is the sum of the polynomials stored in `polys.txt`;
- output the polynomial, in normal form, that is the product of the polynomials stored in `polys.txt`;
- output whether or not the first polynomial in `polys.txt` is divisible by the second polynomial in `polys.txt`, and output the result of the division if the answer is yes.

The procedures in your program should manipulate the polynomials using the type `Polynomial` itself.

Here are the expected outputs of your program for the file test given above.

```
$ a.out polys.txt
The polynomials are:
    x^4 + 8x^2 + 1
    x^14 + x^12 + x^4 + 2
The sum of the polynomials is:
    x^14 + x^12 + 2x^4 + 8x^2 + 3
The product of the polynomials is:
    x^18 + 9x^16 + 9x^14 + x^12 + x^8 + 8x^6 + 3x^4 + 16x^2 + 2
The first polynomial is not divisible by the second one.
```

For another example, suppose that `polys.txt` consists of:

```
3x^7 + 3x^5 + 6x^2 + 6
3x^2 + 3
```

Then the output should be:

```
The polynomials are:
    3x^7 + 3x^5 + 6x^2 + 6
    3x^2 + 3
The sum of the polynomials is:
    3x^7 + 3x^5 + 9x^2 + 9
The product of the polynomials is:
    9x^9 + 18x^7 + 9x^5 + 18x^4 + 36x^2 + 18
The quotient of the first polynomial by the second one is:
    x^5 + 2
```

For still another example, suppose that `polys.txt` consists of:

```
8x^2 + +2x^4 - -5x^3 + +0x^7 + 9x + 4x^0
    2x^2 + +1x + 4
```

Then the output should be:

```
The polynomials are:
    2x^4 + 5x^3 + 8x^2 + 9x + 4
    2x^2 + x + 4
The sum of the polynomials is:
    2x^4 + 5x^3 + 10x^2 + 10x + 8
The product of the polynomials is:
    4x^6 + 12x^5 + 29x^4 + 46x^3 + 49x^2 + 40x + 16
The quotient of the first polynomial by the second one is:
    x^2 + 2x + 1
```

2 The Mandelbrot set

The *Mandelbrot set* is the set of all points c in the complex plane such that the sequence of complex numbers defined as $z_{n+1} = z_n^2 + c$ does not tend to infinity. An approximation is given by the set of all points c such that for all $n \leq 50$, z_n remains in the disk D centered at 0 and of radius 4. Moreover, we will consider only points $c = x + iy$ with $x \in \{-2, 0.5\}$ and $y \in \{-1, 1\}$. When the point under consideration determines a sequence that remains within D after 50 iterations, then a black pixel will be printed out; otherwise a white pixel will be printed out.

The output will be saved as a binary file with the name `mandelbrot.pnm`, the `.pnm` extension indicating that it uses the *Portable Pixel Map* format. Such a file contains 4 lines.

- The first line consists of the string `P6`.
- The second line consists of two integers separated by a space that represent the width and the height of the picture, in pixels, respectively—here we will use 256 for both.
- The third line consists of the maximal color value—here we will use 255.
- The fourth line consists of a sequence of 3 consecutive bytes that represent the level of red, green and blue of each pixel in the image, considered row by row, from left to right; hence this line with contain $3 \times 256 \times 256$ bytes.

To represent a color, use a structure of the form

```
typedef struct { unsigned char r, g, b ; } colour;
```

where `r`, `g` and `b` represent the levels of red, green and blue, respectively, with all values set to `0` for black, and all values set to `255` for white.

The resulting picture can be displayed with the `gimp` application, by executing

```
gimp mandelbrot.pnm
```

The file can also be converted to other formats, for instance to a pdf file, with the command

```
convert -format "pdf" mandelbrot.pnm mandelbrot.pdf
```

and `mandelbrot.pdf` can then be displayed with an application such as Acrobat Reader.

The result will look as follows: