Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2016

Homework 5 – Operator overloading for polynomial class

Due: 13/04/2016, Wednesday, 21:00 (Firm Deadline, NOT to be postponed)

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to implement a class for *univariate polynomials* (polynomials with one variable) and overload some operators. The details of these operators will be given below. Our focus in this homework is on the class design and implementation. The usage of this class in the main function is given to you.

A *univariate polynomial* is mathematically defined as follows.

$$P(x) = a_n.x^n + a_{n-1}.x^{n-1} + \dots + a_0.x^0$$

where a_i, $0 \le i \le n$, are the *coefficients* and n is the *order* of the polynomial. These coefficients might be any real number, but the highest degree coefficient, a_n, must be nonzero. The only exception of this rule is the zero polynomial, which is defined as $P(x) = 0$. In zero polynomial, the degree is zero and the highest degree coefficient, a_0, is also zero.

For examples, the followings are polynomials of degree 5.

$$9x^5 + 3x^4 - x^3 + 5x^2 + 7x + 4$$
$$x^5 - 5x^4 + 2x$$

In this homework, you are going to use polynomial summation, scalar division and derivative. Therefore, we give mathematical definitions for these operations below.

Let $P_1(x)$ and $P_2(x)$ be two polynomials defined as

$$P_1(x) = a_{n1}.x^{n1} + a_{n1-1}.x^{n1-1} + \dots + a_0.x^0$$

$$P_2(x) = b_{n2}.x^{n2} + b_{n2-1}.x^{n2-1} + \dots + b_0.x^0$$

where a_i, $0 \le i \le n1$ and b_j, $0 \le j \le n2$. Then, the resulting polynomial for summation is given as follows.

$$P_1(x) + P_2(x) = \begin{cases} \sum_{k=n2+1}^{n1} a_k.x^k + \sum_{k=0}^{n2}(a_k + b_k).x^k & \text{if } n1 > n2 \\ \sum_{k=n1+1}^{n2} b_k.x^k + \sum_{k=0}^{n1}(a_k + b_k).x^k & \text{if } n1 < n2 \\ \sum_{k=0}^{n1}(a_k + b_k).x^k & \text{if } n1 = n2 \end{cases}$$

Division of the polynomial with a scalar real number k in the form of $\frac{P_1(x)}{k}$ is as follows.

$$\frac{P_1(x)}{k} = \sum_{i=0}^{n1} \frac{a_i.x^i}{k}$$

The derivative of the polynomial $P_1(x)$, which is mathematically denoted as $P'_1(x)$, is defined as follows.

$$P'_1(x) = n1.a_{n1}.x^{n1-1} + (n1-1).a_{n1-1}.x^{n1-2} + \cdots + 2.a_2.x + a_1$$

alternatively,

$$P'_1(x) = \sum_{i=1}^{n1} i.a_i.x^{i-1}.$$

Polynomial Class Design

You have to keep the *order* and the *coefficients* of the polynomial as private data members of the class. Order is a regular integer. Coefficients must be kept as a dynamic array of real numbers (thus, you **must** allocate enough memory for the array dynamically in constructor).

Your class implementation must include the following basic functions:

- Default constructor: Creates a polynomial object with order = 0 and allocates enough memory for the coefficient a_0. Do not initialize this coefficient in the constructor.
- Constructor with *order* parameter: Creates a polynomial object with order given as parameter and allocates enough memory for the coefficients. Do not initialize the coefficients in the constructor.
- Destructor: By definition, destructor returns all dynamically allocated memory to the heap.
- Deep copy constructor: Takes a polynomial object as const-reference parameter and creates a new polynomial object as the deep copy of the parameter.

In this homework, you will overload several operators for the polynomial class. These operators and the details of overloading are given below. You can overload operators as member or free functions. However, in order to test your competence in both mechanisms, we request you to have at least four member and four free functions for these operators (the remaining two is up to you).

>> This operator will be overloaded such that it will read the coefficients of a polynomial object from an input stream (`istream`). The coefficient will be read starting with the highest degree towards the lowest one. The function that you will implement for this operator must not display any prompt. Due to its usage in the program, this function must be a free function.

<< This operator will be overloaded such that it will put a polynomial on an output stream (`ostream`) in mathematical format. See the sample runs for the required output format and be aware of the special cases about coefficients with values 0 and 1, and how x^0 and x^1 are handled. Due to its usage in the program, this function must be a free function.

+ This operator will be overloaded such that it will add two polynomials and return the resulting polynomial. The formula for adding two polynomials is given above in the **Introduction** section.

+= This operator will be overloaded such that it will add two polynomials and assign the resulting polynomial to the polynomial object on the left hand-side of the += operator. This function must be implemented in a way to allow cascaded assignments.

= This operator will be overloaded such that it will assign the polynomial object on the right hand-side of the operator to the polynomial object on the left hand-side. This function must be implemented in a way to allow cascaded assignments. Due to C++ language rules, this operator must be a member function (cannot be free function).

/ This operator will be overloaded such that it will divide a polynomial, which is the left-hand-side operand, with a scalar `double` value, which is the right-hand-side operand. The formula for dividing a polynomial with a scalar in this form is given above in the Introduction section.

> This operator will be overloaded such that it will compare the orders of two polynomial objects. The operator returns true when the order of the polynomial object on the left hand-side is greater than the order of the polynomial on the right hand-side. Otherwise, the operator returns false.

< This operator will be overloaded such that it will compare the orders of two polynomial objects. The operator returns true when the order of the polynomial object on the left hand-side is less than the order of the polynomial on the right hand-side. Otherwise, the operator returns false.

`()` This operator will be overloaded such that it will evaluate the value of the polynomial at a given x value, which is a parameter of type `double`, and return it again as `double`. This operator is referred as `()` in definition but used with an argument in between `(` and `)` in the program. For example, to get the value of a polynomial `myPoly` at $x = 5.4$, use `myPoly(5.4)` in the program. Due to C++ language rules, this operator must be a member function (cannot be free function) and the polynomial to be evaluated is the object on which the operator is called.

~ This operator will be overloaded such that it will take the derivative of a polynomial and return the resulting derivative polynomial. This operator is also a unary operator, taking one polynomial operand. If implemented as a member function, this operand is the object on which this operator is called. If implemented as a free function, it should take one polynomial parameter. In either case, to find the derivative of polynomial `p1`, use `~p1` in the program. We know that this is not the correct mathematical symbol for derivative, but since we cannot overload `'` character in C++, we had to find an alternative notation ☺.

CAUTION: By definition, the highest degree coefficient of a polynomial must be non-zero with the exception of zero polynomial ($P(x) = 0$). After reading a polynomial or after an operation, the highest degree coefficient may become zero. You have to handle these cases appropriately.

In order the see the usage and the effects of these operators, please see **sample runs** and the provided **main.cpp**.

In this homework, you are allowed to implement some other helper member functions, accessors and mutators, if needed. However, you are **not** allowed to use friend functions and friend classes.

You have to have separate header and implementation files, in addition to the main.cpp.

Please do not make use any standard or non-standard *container* or *vector* classes in your homework; you will implement your own class from scratch.

main.cpp

In this homework, **main.cpp** file is given to you within this homework package and we will test your codes with this main function with different inputs. You are not allowed to make any modifications in the main function (of course, you can edit the file to add `#include`s, etc. to the beginning of the file). Inputs are explained with appropriate prompts so that you do not get confused. We strongly recommend you to examine main.cpp file and sample runs before starting your homework.

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark. Especially try different polynomials (other than the ones given in the sample runs) and extreme cases.

Sample Run 1:

```
Enter the order of the first polynomial (p1)
3

Enter the coefficients for polynomial p1 (starting from the coefficient of the highest order term)
-1 -2 3 4

The polynomial p1 is:
-x^3 -2x^2 +3x +4

Enter the order of the second polynomial (p2)
3

Enter the coefficients for polynomial p2 (starting from the coefficient of the highest order term)
1 2 7 8

The polynomial p2 is:
x^3 +2x^2 +7x +8

Enter the orders of the third (p3) and fourth (p4) polynomials
3 2

Enter the coefficients for polynomials p3 and p4 (starting from the coefficient of the highest order term)
1 0 -1 4
-1 3 0

The polynomial p3 and p4 are:
x^3 -x +4
-x^2 +3x

Polynomial p2 after the operation p2 = p2 is:
x^3 +2x^2 +7x +8

Polynomial p5 after the operation p5 = p1 + p2:
```

```
10x +12

The order of p4 is greater than the order of p5

Polynomial p6 after created by copy constructor Polynomial p6 = Polynomial(p1 / 2.5) is:
-0.4x^3 -0.8x^2 +1.2x +1.6

Polynomial p1 after the operation p1 += p3:
-2x^2 +2x +8

Polynomial p5 after the operation p5 = p1 / 2 is:
-x^2 +x +4

Polynomial p3 after the operation p3 = p5 / 5 + p1 / 4:
-0.7x^2 +0.7x +2.8

Polynomial p7 after the operation p7 = p3 + p5 + p6:
-0.4x^3 -2.5x^2 +2.9x +8.4

After the operation p2 = p3 += p1 += p4:
Polynomial p1: -3x^2 +5x +8
Polynomial p3: -3.7x^2 +5.7x +10.8
Polynomial p2: -3.7x^2 +5.7x +10.8
Enter an x value to find p1(x) (corresponding value of p1 at x)
1

p1(x) = 10

Enter an x value to find p5(x) (corresponding value of p5 at x)
9

p5(x) = -68

Polynomial p8 after the operation p8 = ~p1:
-6x +5

Polynomial p4 after the operation p4 = ~p2 + ~p6:
-1.2x^2 -9x +6.9

Press any key to continue . . .
```

Sample Run 2:

```
Enter the order of the first polynomial (p1)
1

Enter the coefficients for polynomial p1 (starting from the coefficient of the highest order term)
2 5

The polynomial p1 is:
2x +5

Enter the order of the second polynomial (p2)
2

Enter the coefficients for polynomial p2 (starting from the coefficient of the highest order term)
-1 -1 0

The polynomial p2 is:
-x^2 -x

Enter the orders of the third (p3) and fourth (p4) polynomials
1 1

Enter the coefficients for polynomials p3 and p4 (starting from the coefficient of the highest order term)
3 3
-5 7

The polynomial p3 and p4 are:
3x +3
-5x +7

Polynomial p2 after the operation p2 = p2 is:
-x^2 -x

Polynomial p5 after the operation p5 = p1 + p2:
-x^2 +x +5

The order of p4 is smaller than the order of p5
```

```
Polynomial p6 after created by copy constructor Polynomial p6 = Polynomial(p1 / 2.5) is:
0.8x +2

Polynomial p1 after the operation p1 += p3:
5x +8

Polynomial p5 after the operation p5 = p1 / 2 is:
2.5x +4

Polynomial p3 after the operation p3 = p5 / 5 + p1 / 4:
1.75x +2.8

Polynomial p7 after the operation p7 = p3 + p5 + p6:
5.05x +8.8

After the operation p2 = p3 += p1 += p4:
Polynomial p1: 15
Polynomial p3: 1.75x +17.8
Polynomial p2: 1.75x +17.8
Enter an x value to find p1(x) (corresponding value of p1 at x)
3

p1(x) = 15

Enter an x value to find p5(x) (corresponding value of p5 at x)
2.5

p5(x) = 10.25

Polynomial p8 after the operation p8 = ~p1:
0

Polynomial p4 after the operation p4 = ~p2 + ~p6:
2.55

Press any key to continue . . .
```

Sample Run 3:

```
Enter the order of the first polynomial (p1)
5

Enter the coefficients for polynomial p1 (starting from the coefficient of the highest order term)
-1 0 1 2 4 7

The polynomial p1 is:
-x^5 +x^3 +2x^2 +4x +7

Enter the order of the second polynomial (p2)
5

Enter the coefficients for polynomial p2 (starting from the coefficient of the highest order term)
1 0 3 -2 -4 -7

The polynomial p2 is:
x^5 +3x^3 -2x^2 -4x -7

Enter the orders of the third (p3) and fourth (p4) polynomials
5 3

Enter the coefficients for polynomials p3 and p4 (starting from the coefficient of the highest order term)
-3.7 0 0 0 0 0
0 0 0 1

The polynomial p3 and p4 are:
-3.7x^5
1

Polynomial p2 after the operation p2 = p2 is:
x^5 +3x^3 -2x^2 -4x -7

Polynomial p5 after the operation p5 = p1 + p2:
4x^3

The order of p4 is smaller than the order of p5

Polynomial p6 after created by copy constructor Polynomial p6 = Polynomial(p1 / 2.5) is:
-0.4x^5 +0.4x^3 +0.8x^2 +1.6x +2.8

Polynomial p1 after the operation p1 += p3:
```

```
-4.7x^5 +x^3 +2x^2 +4x +7

Polynomial p5 after the operation p5 = p1 / 2 is:
-2.35x^5 +0.5x^3 +x^2 +2x +3.5

Polynomial p3 after the operation p3 = p5 / 5 + p1 / 4:
-1.645x^5 +0.35x^3 +0.7x^2 +1.4x +2.45

Polynomial p7 after the operation p7 = p3 + p5 + p6:
-4.395x^5 +1.25x^3 +2.5x^2 +5x +8.75

After the operation p2 = p3 += p1 += p4:
Polynomial p1: -4.7x^5 +x^3 +2x^2 +4x +8
Polynomial p3: -6.345x^5 +1.35x^3 +2.7x^2 +5.4x +10.45
Polynomial p2: -6.345x^5 +1.35x^3 +2.7x^2 +5.4x +10.45

Enter an x value to find p1(x) (corresponding value of p1 at x)
2.1

p1(x) = -157.472

Enter an x value to find p5(x) (corresponding value of p5 at x)
0

p5(x) = 3.5

Polynomial p8 after the operation p8 = ~p1:
-23.5x^4 +3x^2 +4x +4

Polynomial p4 after the operation p4 = ~p2 + ~p6:
-33.725x^4 +5.25x^2 +7x +7

Press any key to continue . . .
```

Sample Run 4:

```
Enter the order of the first polynomial (p1)
3

Enter the coefficients for polynomial p1 (starting from the coefficient of the highest order term)
-1 0 -3.5 2

The polynomial p1 is:
-x^3 -3.5x +2

Enter the order of the second polynomial (p2)
3

Enter the coefficients for polynomial p2 (starting from the coefficient of the highest order term)
1 0 3.5 -2

The polynomial p2 is:
x^3 +3.5x -2

Enter the orders of the third (p3) and fourth (p4) polynomials
2 3

Enter the coefficients for polynomials p3 and p4 (starting from the coefficient of the highest order term)
0 0 0
0 0 -1 1

The polynomial p3 and p4 are:
0
-x +1

Polynomial p2 after the operation p2 = p2 is:
x^3 +3.5x -2

Polynomial p5 after the operation p5 = p1 + p2:
0

The order of p4 is greater than the order of p5

Polynomial p6 after created by copy constructor Polynomial p6 = Polynomial(p1 / 2.5) is:
-0.4x^3 -1.4x +0.8

Polynomial p1 after the operation p1 += p3:
-x^3 -3.5x +2

Polynomial p5 after the operation p5 = p1 / 2 is:
```

```
-0.5x^3 -1.75x +1

Polynomial p3 after the operation p3 = p5 / 5 + p1 / 4:
-0.35x^3 -1.225x +0.7

Polynomial p7 after the operation p7 = p3 + p5 + p6:
-1.25x^3 -4.375x +2.5

After the operation p2 = p3 += p1 += p4:
Polynomial p1: -x^3 -4.5x +3
Polynomial p3: -1.35x^3 -5.725x +3.7
Polynomial p2: -1.35x^3 -5.725x +3.7
Enter an x value to find p1(x) (corresponding value of p1 at x)
-1

p1(x) = 8.5

Enter an x value to find p5(x) (corresponding value of p5 at x)
-1

p5(x) = 3.25

Polynomial p8 after the operation p8 = ~p1:
-3x^2 -4.5

Polynomial p4 after the operation p4 = ~p2 + ~p6:
-5.25x^2 -7.125

Press any key to continue . . .
```

Sample Run 5:

```
Enter the order of the first polynomial (p1)
3

Enter the coefficients for polynomial p1 (starting from the coefficient of the highest order term)
1 -1 1 1

The polynomial p1 is:
x^3 -x^2 +x +1

Enter the order of the second polynomial (p2)
3

Enter the coefficients for polynomial p2 (starting from the coefficient of the highest order term)
1 1 1 -1

The polynomial p2 is:
x^3 +x^2 +x -1

Enter the orders of the third (p3) and fourth (p4) polynomials
4 3

Enter the coefficients for polynomials p3 and p4 (starting from the coefficient of the highest order term)
1 0 1 0 0
0 1 0 1

The polynomial p3 and p4 are:
x^4 +x^2
x^2 +1

Polynomial p2 after the operation p2 = p2 is:
x^3 +x^2 +x -1

Polynomial p5 after the operation p5 = p1 + p2:
2x^3 +2x

The order of p4 is smaller than the order of p5

Polynomial p6 after created by copy constructor Polynomial p6 = Polynomial(p1 / 2.5) is:
0.4x^3 -0.4x^2 +0.4x +0.4

Polynomial p1 after the operation p1 += p3:
x^4 +x^3 +x +1

Polynomial p5 after the operation p5 = p1 / 2 is:
0.5x^4 +0.5x^3 +0.5x +0.5

Polynomial p3 after the operation p3 = p5 / 5 + p1 / 4:
0.35x^4 +0.35x^3 +0.35x +0.35
```

```
Polynomial p7 after the operation p7 = p3 + p5 + p6:
0.85x^4 +1.25x^3 -0.4x^2 +1.25x +1.25

After the operation p2 = p3 += p1 += p4:
Polynomial p1: x^4 +x^3 +x^2 +x +2
Polynomial p3: 1.35x^4 +1.35x^3 +x^2 +1.35x +2.35
Polynomial p2: 1.35x^4 +1.35x^3 +x^2 +1.35x +2.35
Enter an x value to find p1(x) (corresponding value of p1 at x)
1

p1(x) = 6

Enter an x value to find p5(x) (corresponding value of p5 at x)
-1

p5(x) = 0

Polynomial p8 after the operation p8 = ~p1:
4x^3 +3x^2 +2x +1

Polynomial p4 after the operation p4 = ~p2 + ~p6:
5.4x^3 +5.25x^2 +1.2x +1.75

Press any key to continue . . .
```

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate. Since using classes is mandated in this homework, a proper object oriented design and implementation, and following the rules mentioned in this homework specification will also be considered in grading.

Since you will use dynamic memory allocation in this homework, it is very crucial to properly manage the allocated area and return the deleted parts to the heap whenever appropriate. Inefficient use of memory may reduce your grade.

When we grade your homework we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we are going to run your programs in *Release* mode and **we may test your programs with large test cases**. Of course, your program should work in *Debug* mode as well.

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade. Name your solution, project, cpp file that contains your main program using the following convention (the necessary file extensions such as .sln, .cpp, etc, are to be added to it):

“SUCourseUserName_YourLastname_YourName_HWnumber”

Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw5

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case add informative phrases after the hw number. However, do not add any other character or phrase to the file names.

Now let us explain which files will be included in the submitted package. Visual Studio 2012 will create two *debug* folders, one for the solution and the other one for the project. You should delete these two *debug* folders. Moreover, if you have run your program in release mode, Visual Studio may create *release* folders; you should delete these as well. Apart from these, Visual Studio 2012 creates a file extension of *.sdf* ; you will also delete this file. The remaining content of your solution folder is to be submitted after compression. Compress your solution and project folders using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all of the solution, project and source code files that belong to the latest version of your homework. Especially double-check that the zip file contains your cpp and (if any) header files that you wrote for the homework.

Moreover, we strongly recommend you to check whether your zip file will open up and run correctly. To do so, unzip your zip file to another location. Then, open your solution by clicking the file that has a file extension of .sln. Clean, build and run the solution; if there is no problem, you could submit your zip file. Please note that the deleted files/folders may be regenerated after you build and run your program; this is normal, but do not include them in the submitted zip file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw5.zip is a valid name, but

Hw5_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Albert Levi, Ömer Mert Candan