CPP Problem Design

Subject: Design Polynomial Class

Contributor: 謝公耀, 陳俊儒, 廖宣瑋 edit : 葉定豪

Main testing concept:

Basics	Functions
■ C++ BASICS	□ SEPARATE COMPILATION AND NAMESPACES
□ FLOW OF CONTROL	□ STREAMS AND FILE I/O
□ FUNCTION BASICS	□ RECURSION
□ PARAMETERS AND OVERLOADING	□ INHERITANCE
■ ARRAYS	□ POLYMORPHISM AND VIRTUAL FUNCTIONS
■ STRUCTURES AND CLASSES	□ TEMPLATES
■ CONSTRUCTORS AND OTHER TOOLS	□ LINKED DATA STRUCTURES
■ OPERATOR OVERLOADING, FRIENDS,AND REFERENCES	□ EXCEPTION HANDLING
□ STRINGS	□ STANDARD TEMPLATE LIBRARY
■ POINTERS AND DYNAMIC ARRAYS	□ PATTERNS AND UML

Description:

Please implement a class called **Polynomial** to handle one-dimensional polynomials. This class needs to be able to store the coefficients and implement operations such as addition, subtraction, multiplication, and assignment.

An example of a polynomial of a single variable, x, is $x^3 + 3x^2 + 7x + 8$, which can be expressed with a set of coefficients like **{8, 7, 3, 1}**.

- Please design your own data structure to store these polynomials and implement the following methods:
  - **Polynomial()**
    Construct a zero polynomial.
  - **Polynomial(double* param, int size)**
    Construct a one-dimensional polynomial based on the given coefficients which have the given size.
  - **Polynomial(const Polynomial& poly)**
    Copy constructor.
- Suppose there were three polynomials: ***poly1(3x + 9)***, ***poly2(0 x^3+5x^2 + 6x + 8)*** and ***poly()***.
  - **int mySize()**
    Return the number of terms of the polynomial.
    For Example: *poly1.mySize()* should return 2 and *poly2.mySize()* should be 3(first non-zero coefficient).
  - **double evaluate(const Polynomial& poly, const double& var)**
    Return the value of the polynomial after substituting *var* into the variables.
    For Example: *evaluate(poly1, 2)* should return 15.
- Overload operators to meet the following operational requirements.
  - **Assignment:**

Assign the value from a polynomial to another polynomial. (define operator =)
For Example: *poly* = *poly1*; Then *poly* is 3x+9.

- **Return the coefficient** of the certain power variable in the polynomial. (define operator [])
  For example:
  poly1[0] needs to return the coefficient of x to the power of 0, which has the value of 9.
  poly1[2] = 1, then poly1 becomes $x^2 + 3x + 9$
  Index will always be positive.

- **Addition:**
  Implement the addition of two polynomials or a polynomial and a constant number. (define operator +)
  For Example: *poly* = *poly1* + *poly2*; Then *poly* is $5x^2 + 9x + 17$.
  poly = *5* + *poly1*; Then *poly* is $3x + 14$.
  poly = *poly1* + *10.5*; Then *poly* is $3x + 19.5$.

- **Subtraction:**
  Implement the subtraction of two polynomials or a polynomial and a constant number. (define operator −)
  For Example: *poly* = *poly1* - *poly2*; Then *poly* is $-5x^2 - 3x + 1$.
  poly = *6* - *poly1*; Then *poly* is $-3x - 3$.
  poly = *poly1* - *1.6*; Then *poly* is $3x + 7.4$.

- **Multiplication:**
  Implement the multiplication of two polynomials or a polynomial and a constant number. (define operator *)
  For Example: *poly* = *poly1* * *poly2*; Then *poly* is $15x^3 + 63x^2 + 78x + 72$.
  poly = *23* * *poly1*; Then *poly* is $69x + 207$.
  poly = *poly1* * *7*; Then *poly* is $21x + 63$.

- This exercise will not provide the template program. Please design the functionality required by the topic on your own.

Input:
No inputs.
**The main() function in your submission will be replaced when judging.
**You can use the main() function in "Other Notes" to test your program.

Output:
The result of executing your program with the given main function.

Sample Input / Output：

Sample Input	Sample Output
No inputs	Polynomial q term with degree 0 has coefficient 3 term with degree 1 has coefficient 2 term with degree 2 has coefficient 1 Polynomial c term with degree 0 has coefficient 1

	term with degree 1 has coefficient 2 term with degree 2 has coefficient 0 term with degree 3 has coefficient 3 value of q(2) is 11 value of p(2) is 11 value of r(2) is 29 value of c(2) is 29 value of (q + c)(2) is 40 value of (q - c)(2) is -18 size of q*c is 6 Polynomial r (= q*c) term with degree 0 has coefficient 3 term with degree 1 has coefficient 8 term with degree 2 has coefficient 5 term with degree 3 has coefficient 11 term with degree 4 has coefficient 6 term with degree 5 has coefficient 3 value of (q * c)(2) is 319

□ **Easy, only basic programming syntax and structure are required.**

■ **Medium, multiple programming grammars and structures are required.**

□ **Hard, need to use multiple program structures or complex data types.**

Expected solving time:

40 minutes

Other notes:

```
int main()
{
	Polynomial empty;
	double one[] = { 1 };
	Polynomial One(one, 1);
	double quad[] = { 3, 2, 1 };
	double cubic[] = { 1, 2, 0, 3 };
	Polynomial q(quad, 3); // q is 3 + 2*x + x*x
	Polynomial c(cubic, 4);// c is 1 + 2*x + 0*x*x + 3*x*x*x
	Polynomial p = q;  // test copy constructor
	Polynomial r;
	r = q;		//test operator=
	r = c;

	cout << "Polynomial q " << endl;
	for (int i = 0; i < 3; i++)
		cout << "term with degree " << i << " has coefficient " << q[i] << endl;

	cout << "Polynomial c " << endl;
	for (int i = 0; i < 4; i++)
		cout << "term with degree " << i << " has coefficient " << c[i] << endl;

	cout << "value of q(2) is " << evaluate(q, 2) << endl;
	cout << "value of p(2) is " << evaluate(p, 2) << endl;
	cout << "value of r(2) is " << evaluate(r, 2) << endl;
	cout << "value of c(2) is " << evaluate(c, 2) << endl;

	r = q + c;
	cout << "value of (q + c)(2) is " << evaluate(r, 2) << endl;
```

```
	r = q - c;
	cout << "value of (q - c)(2) is " << evaluate(r, 2) << endl;

	r = q * c;
	cout << "size of q*c is " << r.mySize() << endl;
	cout << "Polynomial r (= q*c) " << endl;

	for (int i = 0; i < r.mySize(); i++)
		cout << "term with degree " << i << " has coefficient " << r[i] << endl;

	cout << "value of (q * c)(2) is " << evaluate(r, 2) << endl;
	return 0;
}
```