CPP Problem Design

Subject: Design Polynomial Class

Contributor: 謝公耀, 陳俊儒, 廖宣瑋

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:

Using dynamic arrays, implement a polynomial class with polynomial addition, subtraction, and multiplication. Remarks: A variable in a polynomial does nothing but act as a placeholder for the coefficients. Hence, the only interesting thing about polynomials is the array of coefficients and the corresponding exponent. Think about the polynomial

x*x*x + x + 1

Where is the term in x*x? One simple way to implement the polynomial class is to use an array of doubles to store the coefficients. The index of the array is the exponent of the corresponding term. If a term is missing, then it simply has a zero coefficient. There are techniques for representing polynomials of high degree with many missing terms. These use so-called sparse matrix techniques. Unless you already know these techniques, or learn very quickly, do not use these techniques.

Please

(1) provide a default constructor, a copy constructor, and a parameterized constructor that enables an arbitrary polynomial to be constructed.

(2) Supply an overloaded operator = and a destructor.

(3) Provide these operations: polynomial + polynomial, constant + polynomial, polynomial + constant, polynomial - polynomial, constant - polynomial, polynomial - constant. polynomial * polynomial, constant * polynomial, polynomial * constant,

(4) Supply functions to assign and extract coefficients, indexed by exponent.

(5) Supply a function to evaluate the polynomial at a value of type double.

You should decide whether to implement these functions as members, friends, or standalone functions.

NOTES:

The default constructor creates an empty polynomial. A zero polynomial has degree 0, since it has only the zero degree coefficient.

In the coefficient array, the index is the value of the exponent of term having this coefficient. For example, the index 0 entry is the constant coefficient, the index 1 entry is coefficient of the linear term, the index 2 entry is the coefficient of the quadratic term (term in x2), etc.

The size of the coefficient array include a degree 0 entry, so the size is the degree of the polynomial + 1.

Input:

main.in

Output:

Sample out

Sample Input / Output：

Sample Input	Sample Output
main.in	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

☐ Eazy,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: