Java Homework Help.

Java Homework Help

C ++ can define custom operators for their own data types. The operator is defined as the usual member function of the class, only after determining the returned type is set by the operator keyword.

An example of the definition of the addition operator:

INT Operator + (int value) {Return Number + Value; }

The operator can be unary or binary. The unary operator does not accept any arguments. For example, the denying operator is "!". The binary operator accepts an additional parameter. For example, in the case of addiction, the second term is accepted.

To clarify the picture, try to write a SIMPLE_FRACTION class, which will describe a simple fraction with an integer numerator and denominator. And we define the operators of addition, subtraction, multiplication and divisions for this class.

/ * * Class describing a simple fraction * / class simple_fraction {public: simple_fraction (int numerator, int denominator) {if (Denominator == 0) // Error divisions on zero Throw Std :: Runtime_error ("Zero Division Error"); this-> numerator = numerator; This-> Denominator = Denominator; } // Definition of basic mathematical operations for simple fraction Double Operator + (Int Val) {Return Number () + Val; } // Addition of Double Operator- (int val) {return number () - VAL; } // Subtracting Double Operator * (int val) {Return Number () * Val; } // Multiplying Double Operator / (Int Val) // division {if (val == 0) {Throw Std :: Runtime_error ("Zero Division Error"); } Return Number () / Val; } // Getting a fraction in the form of a conventional double-number Double Number () {Return Numerator / (Double) DENOMINATOR; } Private: int numerator; // Numerator INT DENOMINATOR; // denominator};

For the division operation, we also made a division check on zero.

Example of using the SIMPLE_FRACTION class:

// Simple fraction 2/3 Simple_Fraction FR (2, 3); Double Sum = FR + 10; // The sum DOUBLE DIFF = FR - 10; // Difference Double Factor = FR * 10; // Product Double Div = FR / 10; // Private

Operators can be overloaded the same as conventional class member functions. For example, you can overload the addition operator for two simple fractions, which will return a new simple fraction. Then, we will have to bring a fraction to a common denominator and return another simple fraction.

Task: Improve the SIMPLE_FRACTION class. Overload the addition operators, subtraction, multiplication and division so that you can perform operations on two simple fractions and get a new simple fraction. Implement bringing two fractions to a common denominator.

An example of using the future class:

SIMPLE_FRACTION FR1 (2, 3); SIMPLE_FRACTION FR2 (3, 4); // 2/3 + 3/4 is 17/12
Simple_Fraction SUM = FR1 + FR2;