PROGRAMMING ASSIGNMENT.

programming assignment

Class methods can also be overloaded as conventional functions. This is especially convenient when you need to make several constructors that will take different parameters.

For example, let's try to create the basis of the Decimal class, which sells long arithmetic for arbitrary accuracy. In such cases, the number is usually stored inside the string, and the logic of mathematical operations is implemented through the writing of the corresponding class operators.

We will make it so that in the designer of this class it was possible to transfer both a string and number of the type of double.

// Transfer to Decimal Num1 string constructor ("10000000.999999"); // Transfer of the number of Decimal Num2 (10000.0);

In order for the class to support such versatility, we will make two different constructors for string and numbers:

/ ** * Imagine that this class sells long arithmetic for numbers of any * accuracy * / class decimal {public: / * * Designer, which accepts a line containing the number * / Decimal (String Number) {CLOG << "FIRST CONSTRUCTOR CALLED \ N "; } / ** * Designer assigned the number of type Double * / Decimal (Double Number) {Clog << "Second Constructor Called \ n"; } Private: String Number; };

When passing a line, the first constructor will be caused, and when the number is transmitted - the second.

Full text of the program:

#include <iostream> #include <String> Using Namespace STD; / ** * Imagine that this class sells long arithmetic for numbers of any accuracy * / class decimal {public: / * * Designer receiving a string argument, * containing the number * / Decimal (String Number) {CLOG << "FIRST CONSTRUCTOR CALLED \ N "; this-> number = number; } / ** * The designer receives the number of type Double * / Decimal (Double Number) {/ ** * We convert the double to the string with the highest possible accuracy * and write the resulting value in this-> Number * / CLOG << "Second Constructor Called \ n "; } Private: String Number; }; INT MAIN () {// The first Designer Decimal Num1 will be called ("10000000.999999"); // will be caused by the second Designer Decimal Num2 (10000.0); CIN.GET (); Return 0; }

Of course, our class does nothing yet, because the implementation of long arithmetic is beyond the

scope of this article. But on this example, it can be understood when it can be useful to use the overload of class methods.

Task: Try to write a class Student, in the designer of which it will be possible to transmit either his name and surname or the name and year of birth. When transferring a year of birth, the approximate age of the student should be considered. Example of using class:

Student Stud1 ("Ivan", "Ivanov"); Student Stud2 (Ivan, 1990);

The following lesson is the definition and overload of operators in C ++.