Перегрузка операторов позволяет определить действия, которые будет выполнять оператор. Перегрузить можно только те операторы, которые уже определены в C++. Создать новые операторы нельзя.

Синтаксис перегрузки операторов очень похож на определение функции с именем operator@, где @ — это идентификатор оператора (например +, -, <<, >>). Рассмотрим простейший пример:

class Integer   
{   
 private:   
 int value;   
 public:   
 Integer(int i): value(i) {}   
 const Integer operator+(const Integer& rv) const   
 {   
 return (value + rv.value);   
 }   
};

В данном случае, оператор оформлен как член класса, аргумент определяет значение, находящееся в правой части оператора. Существует два основных способа перегрузки операторов: через дружественные функции или методы класса.

Если функция оператора определена как отдельная функция и не является членом класса, то количество параметров такой функции совпадает с количеством операндов оператора. Например, у функции, которая представляет унарный оператор, будет один параметр, а у функции, которая представляет бинарный оператор, - два параметра. Если оператор принимает два операнда, то первый операнд передается первому параметру функции, а второй операнд - второму параметру. При этом как минимум один из параметров должен представлять тип класса

**Не всё может быть перегружено через дружественные функции**

Операторы присваивания (=), индекса ([]), вызова функции (()) и выбора члена (->) перегружаются через методы класса — это требование языка C++.

**Не всё может быть перегружено через методы класса**

Операции ввода/вывода, потому что при перегрузке через метод класса в качестве левого операнда используется текущий объекта, а в этом случае левым операндом является объект типа std::istream/std::ostream.

**Операторы, которые нельзя перегружать:**

?: | :: | . | .\* | sizeof | typeid и различные cast операторы