1. Может возникнуть ситуация, когда есть класс с данными Private и функция, которая работает с этим классом, но которая не находится в его теле. Например, есть класс, в котором хранятся данные, и функция (или другой класс), которая выводит эти данные на экран. В таком случае могут пригодится дружественные классы и функции
2. Дружественные функции нарушают принцип инкапсуляции, поэтому по возможности стоит их избегать. Правила описания дружественной функции:

* Друж. Функция объявляется внутри класса, к элементам которого ей нужен доступ, с ключевым словом friend.
* Друж. Функция может быть обычной функцией или методом другого ранее определенного класса
* Одна функция может быть дружественной сразу нескольким классам

1. Унарную операцию можно перегрузить: 1) как компонентную функцию класса; 2) как внешнюю (глобальную) функцию
2. Операндом является вызвавший функцию объект
3. Один параметр типа класса
4. А
5. Два параметра типа класса
6. Постфиксные формы операторов возвращают исходное значение, а префиксные – новое
7. Оператор присваивания может быть перегружен для конкретного класса, например, так:

CMyClass operator=(CMyClass mc);

CMyClass CMyClass::operator=(CMyClass mc)

{

};

1. Значение, помимо копирования этого значения в переменную
2. Перегрузить можно оператор вывода в поток ostream, что позволит использовать экземпляры класса в строке потока вывода с таким же синтаксисом, как и для встроенных типов.

При этом в качестве левого операнда оператор << должен иметь неконстантную ссылку на поток ostream, а значит данный оператор всегда определяется внешней функцией, а не методом класса.

В качестве результата следует возвращать исходную ссылку на поток.

1. Будет возвращено обновленное значение s
2. Прибавит единицу к s