17. Перегрузка операторов. Дружественные отношения классов. Поиск имен в области видимости классов и дружественных функций.

Перегрузка операторов

Ахтунг: информация взята с Метанита, т.к. информации об этом я НЕ нашёл в лекциях Чабанова

Перегрузка операторов (operator overloading) позволяет определить для объектов классов втроенные операторы, такие как +, -, \* и т.д. Для определения оператора для объектов своего класса, необходимо определить функцию, название которой содержит слово operator и символ перегружаемого оператора. Функция оператора может быть определена как член класса, либо вне класса.

Перегрузить можно только те операторы, которые уже определены в C++. Создать новые операторы нельзя. Также нельзя изменить количество операндов, их ассоциативность, приоритет.

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.

Дружественные отношения классов

Ахтунг: информация взята с лекции Чабанова

В С++ управление доступом реализовано посредством модификаторов доступа и ключевого слова friend.

Ключевое слово может быть применено к функциям (методам класса тоже) и классам. Такие функции и классы называются дружественные. Дружественные функции и классы имеют полный доступ к членам класса в независимости от модификатора доступа этого члена.

Данное ключевое слово нарушает принцип инкапсуляции, так как обычная функция и внешний класс НЕ являются частью рассматриваемого класса.

Чтобы функция или класс стал дружественным, нужно указать объявление внутри целевого класса. То есть, сделать произвольную функцию дружественной, без ведома целевого класса НЕВОЗМОЖНО.

Поиск имен в области видимости классов и дружественных функций

Ахтунг: информация взята из ChatGPT (НЕ ТЕЛЕГРАММ, ОН ХУЙНЯ ЁБАНАЯ, КАК ВЫ ИМ БЛЯТЬ ПОЛЬЗУЕТЕСЬ), т.к. информации по данной теме я НЕ нашёл ни на Метаните, ни в лекциях Чабанова

В C++ поиск имен в области видимости классов и дружественных функций осуществляется с помощью правил, которые определяют, какой именно идентификатор будет выбран при обращении к нему.

Поиск имен в области видимости классов начинается с текущего класса и продолжается вверх по иерархии наследования, пока не будет найдено соответствующее имя. Если идентификатор не найден в текущем классе, поиск продолжается в родительском классе и так далее. Если идентификатор не найден ни в одном из родительских классов, поиск расширяется на глобальную область видимости.

Дружественные функции могут иметь доступ к приватным и защищенным членам класса, но они не являются членами самого класса и не наследуются. При обращении к идентификатору в дружественной функции, поиск начинается с области видимости, в которой определена дружественная функция, и продолжается вверх по области видимости, пока не будет найдено соответствующее имя.

Если идентификатор не найден ни в области видимости класса, ни в области видимости дружественной функции, поиск расширяется на глобальную область видимости.

Вот пример, демонстрирующий поиск имен в области видимости классов и дружественных функций:

