ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

Перевантаження операторів дозволяє перевизначити оператори C # для застосування їх до типів, певним користувачем. Перевантаження операторов назвали "синтаксичним цукром" ("syntactic sugar" '), маючи на увазі, що це лише інший спосіб виклику методу. Це також повинно говорити про те, що ця можливість нічого фундаментального в язик не привносить. Хоча з формальної точки зору це і так, перевантаження операторів пов'язана з одним з найважливіших аспектів ООП - абстракцією.

C # дає можливість будувати спеціальні класи і структури, які можуть унікально реагувати на один і той же набір лексем, наприклад операція +.

Коли оператор перевантажується, жодне з його початкових призначень не втрачається. Він просто виконує ще одну, нову операцію щодо конкретного об'єкта. Тому перевантаження оператора +, наприклад, для обробки зв'язного списку зовсім не впливає на призначення по відношенню до цілих чисел, тобто до додаванню.

Головна перевага перевантаження операторів полягає в тому, що вона дозволяє плавно інтегрувати клас нового типу в середу програмування. Подібного роду розширюваність типів є важливою складовою ефективності такого об'єктно-орієнтованої мови програмування, як C #. Як тільки для класу визначаються оператори, з'являється можливість оперувати об'єктами цього класу, використовуючи звичайний синтаксис виразів в C #. Перевантаження операторів є однією з найсильніших сторін мови C #.

Основи перевантаження операторів

Перевантаження операторів тісно пов'язана з перевантаженням методів. Для перевантаження оператора служить ключове слово operator, що визначає операційний метод, який, в свою чергу, визначає дію оператора щодо свого класу.

У C # існують три види операцій класу: унарні, бінарні і операції перетворення типу.

При описі операцій необхідно дотримуватися таких правил:

• операція повинна бути описана як відкритий статичний метод класу (специфікатор public static);

• параметри в операцію повинні передаватися за значенням;

• сигнатури всіх операцій класу повинні відрізнятися;

• типи, використовувані в операції, повинні мати не менші права доступу, ніж сама операція (тобто повинні бути доступні при ис-користуванні операції).

Існують дві форми операторних методів (operator): одна - для унарних операторів, інша - для бінарних. Нижче наведена загальна форма для кожного різновиду цих методів.

Тут замість op підставляється перевантажується оператор, наприклад + або /; а возвращаемий_тіп позначає конкретний тип значення, що повертається зазначеної операцією. Це значення може бути будь-якого типу, але найчастіше це було визначено такого ж типу, як і у класу, для якого перевантажується оператор. Така кореляція спрощує застосування перевантажуються операторів в виразах. Для унарних операторів операнд позначає передається операнд, а для бінарних операторів те ж саме позначають операнд1 і операнд2. Зверніть увагу на те, що операційні методи

повинні мати обидва типи, public і static.

Тип операнда унарних операторів повинен бути таким же, як і у класу, для якого перевантажується оператор. А в бінарних операторах хоча б один з операндів має бути такого ж типу, як і у його класу. Отже, в C # не допускається перевантаження будь-яких операторів для об'єктів, які ще не були створені. Наприклад, призначення оператора + можна перевизначити для елементів типу int або string. І ще одне зауваження: в параметрах оператора можна використовувати модифікатор ref або out.

Перевантаження бінарних операторів

+, -, *, /,%, &, |, ^, <<, >> Ці бінарні операції можуть бути перевантажені
При заходів. Створимо клас описує точку в тривимірному просторі і перевантажимо для нього оператори складання + і -. Клас буде конструктор для генерації координат і метод для виведення координат.

Додамо до вихідного коду методи для перевантаження операторів

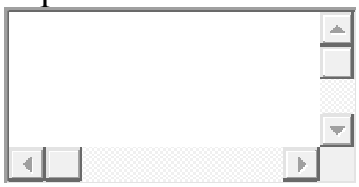
код програми

Уважно проаналізуємо наведену вище програму, починаючи з перегружаемого оператора +. Коли оператор + оперує двома об'єктами типу ThreeD, то величини їх відповідних координат складаються, як показано в оголошенні операційного методу operator + (). Слід, однак, мати на увазі, що цей оператор не видозмінює значення своїх операндів, а лише повертає новий об'єкт типу ThreeD, що містить результат операції додавання координат. Для того щоб стало зрозуміліше, чому операція + не змінює вміст об'єктів, які виступають в ролі її операндів, звернемося до прикладу звичайної операції арифметичного додавання: 10 + 12. Результат цієї операції дорівнює 22, але вона не міняє ні число 10, ні число 12. незважаючи на те що жодне з правил не перешкоджає перевантаженому оператору змінити значення одного зі своїх операндів, все ж краще, щоб дії цього оператора відповідали його прямим призначенням.

Зверніть увагу на те, що метод operator + () повертає об'єкт типу ThreeD. Цей метод міг би повернути значення будь-якого допустимого в C # типу, але завдяки тому що він повертає об'єкт типу ThreeD, оператор + можна використовувати в таких складових виразах, як a + b + c. В даному випадку вираз a + b дає результат типу ThreeD, який можна потім скласти з об'єктом з того ж типу. Якби вираз a + b давало результат іншого типу, то обчислити складене вираз a + b + c було б просто невозможно.Следует також підкреслити, що коли окремі координати точок складаються в операторі operator + (), то в результаті такого додавання виходять цілі значення, оскільки окремі координати x, y і z представлені цілими величинами. Але сама перевантаження оператора +

для об'єктів типу ThreeD не робить ніякого впливу на операцію складання цілих значень, тобто вона не змінює початкове призначення цього оператора.А тепер проаналізуємо операторний метод operator- (). Оператор - діє так само, як і оператор +, але для нього важливий порядок проходження операндів. Нагадаємо, що складання носить комутативними характер (від перестановки доданків сума не змінюється), чого не можна сказати про вирахування: A - B не те ж саме, що і B - A! Для всіх довічних операторів першим параметром операційного методу є лівий операнд, а другим параметром - правий операнд. Тому, реалізуючи перевантажуються варіанти некомутативних операторів, слід пам'ятати, який саме операнддолжен бути вказано зліва і який - справа.Перегрузка унарних операторів +, -,! , ~, ++, -, True, false Ці унарні операції можуть бути перегруженіПрімер. Додамо методи для перевантаження операторів -, ++ В даному прикладі створюється новий об'єкт, в полях якого зберігаються негативні значення операнда перегружаемого унарна оператора, після чого цей об'єкт повертається операційним методом. У C # перевантаження операторів ++ і - здійснюється досить просто. Для цього досить повернути інкрементіровать або декрементірованное значення, але не змінювати викликає об'єкт. А все інше візьме на себе компілятор C #, розрізняючи префіксние і постфіксні форми цих операторів. Код програми Перевантаження операторів true і falseКлючевие слова true і false можна також використовувати в якості унарних операторів для цілей перевантаження. Перенавантажувані варіанти цих операторів дозволяють визначити призначення ключових слів true і false спеціально для створюваних классов.После перевантаження цих ключових слів як унарних операторів для конкретного класу з'являється можливість використовувати об'єкти цього класу для управління операторами if, while, for і do-while або ж в умовному вираженні?. Оператори true і false повинні перевантажуватися попарно, а не окремо. Нижче наведена загальна форма перевантаження цих унарних операторов.public static bool operator true (тіп_параметра операнд) {// Повернення логічного значення true або false.} Public static bool operator false (тіп_параметра операнд) {// Повернення логічного значення true або false.} Зверніть увагу на те, що і в тому і в іншому випадку повертається результат типу bool.Прімер. Напишемо реалізацію методів перевантаження операторів true і false у класі ThreeD. У кожному з цих операторів перевіряється така умова: якщо хоча б одна з координат об'єкту типу ThreeD дорівнює нулю, то цей об'єкт правдивий, а якщо все три його координати дорівнюють нулю, то такий об'єкт хибна. Фрагмент коду Перевантаження операторів отношеніяОператори відносини, наприклад == і <, можуть також перевантажуватися, прічемочень просто. Як правило, перевантажений оператор відносини повертає логічне значення true і false. Це цілком відповідає правилам звичайного застосування подібних операторів і дає можливість використовувати їх перевантажуються різновиди в умовних виразах. Якщо ж повертається результат іншого типу, то тим самим сильно обмежується можливість застосування операторів відносини. Приклад. Напишемо методи в

яких перевантажуються оператори <i>. В даному прикладі ці оператори служать для порівняння об'єктів ThreeD, виходячи з їх відстані до початку координат. Один об'єкт вважається більше іншого, якщо він знаходиться далі від початку координат. А крім того, один об'єкт вважається менше іншого, якщо він знаходиться ближче до початку координат. Такий варіант реалізації дозволяє, зокрема, визначити, яка з двох заданих точок знаходиться на більшій сфері. Якщо ж жоден з операторів не повертає логічне значення true, то обидві точки знаходяться на одній і тій же сфері. Зрозуміло, возможниі інші алгоритми впорядкування. Закомментіруем перевантажені true і false для даного прикладу. Фрагмент коду Перевантаження логічних операторовКак вам повинно бути вже відомо, в C # передбачені наступні логічні оператори: &, |,!, && I ||. З них перевантаження, безумовно, підлягають тільки оператори &, | і!. Якщо не користуватися укороченими логічними операторами, то перевантаження операторів & і | можна виконувати абсолютно природним шляхом, отримуючи в кожному випадку результат типу bool. Аналогічний результат, як правило, дає і перевантажується оператор! .Приклад. Написати методи в яких демонструється перевантаження логічних операторів!, & I | для об'єктів типу Thre



206/5000

eD. Як і в попередньому прикладі, об'єкт типу ThreeD вважається дійсним, якщохоча б одна з його координат не дорівнює нулю. Якщо ж все три координатиоб'єкта дорівнюють нулю, то він вважається хибним. фрагмент коду