#### Пріоритети операторів

Якщо вираз містить кілька операторів, то порядок їх виконання визначає пріоритет категорії, до якої належить оператор (таблиця 2.11).

Таблиця №2.11. Пріоритети операторів: від найвищого до найнижчого

|  |  |
| --- | --- |
| **Категорії операторів Оператори** | |
| **Первинні** | a.x, f(x), a[x], x++, x--, new, typeof, checked, unchecked |
| **Унарні** | +, -, !, ~, ++x, --x, (T)x |
| **Мультиплікативні** | \*, /, % |
| **Аддитивні** | +, - |
| **Зсув** | <<, >> |
| **Відношення і тип** | <, >, <=, >=, is, as |
| **Рівність** | ==, != |
| **Логічне "І"** | & |
| **Логічне "Виключаюче АБО"** | ^ |
| **Логічне "АБО"** | | |
| **Умовне "І"** | && |
| **Умовне "АБО"** | || |
| **Умова** | ? : |
| **Присвоєння** | =, \*=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |= |

При побудові виразу, якщо слід змінити пріоритетність виконання різних його операторів, використовують круглі дужки. Тоді першими обчислюються вирази в дужках, починаючи з найглибшого рівня вкладеності.

#### Прості арифметичні оператори

Для базових арифметичних операцій використовують оператори згідно таблиці 2.12. Їх використання відповідає загальновідомим правилам математики.

Оператор % виконує ділення операндів, але повертає не результат ділення, а залишок. Наприклад, виконання рядка програми

Console.WriteLine( 11 % 3); приведе до виводу на екран числа 2 – залишку від ділення 11 на 3.

Таблиця №2.12. Оператори простих арифметичних операцій

|  |  |
| --- | --- |
| **Оператор** | **Опис** |
| **+** | Додавання |
| **-** | Віднімання |
| **\*** | Множення |
| **/** | Ділення |
| **%** | Залишок при діленні |

#### Оператори для порівняння

Оператори для порівняння порівнюють операнди і повертають результат логічного типу (bool). Їх перелічено та описано у таблиці 2.13.

Таблиця №2.13. Оператори для порівняння

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Назва** | **Опис** |
| **<** | Менше | Повертає true, якщо перший операнд менший за другий, і false в протилежному випадку |
| **>** | Більше | Повертає true, якщо перший операнд більший за другий, і false в протилежному випадку |
| **<=** | Менше або дорівнює | Повертає true, якщо перший операнд менший або дорівнює другому, і false в протилежному випадку |
| **>=** | Більше або дорівнює | Повертає true, якщо перший операнд більший або дорівнює другому, і false в протилежному випадку |
| **==** | Дорівнює | Повертає true, якщо перший операнд дорівнює другому, і false в протилежному випадку |
| **!=** | Не дорівнює | Повертає true, якщо перший операнд не дорівнює другому, і false в протилежному випадку |

#### Оператори інкременту та декременту

Оператори для інкременту (приросту на 1) та декременту (зменшення на 1) приведено у таблиці 2.14. Вони мають дві форми: префіксну та постфіксну. У префіксній формі оператор розміщують перед операндом, а в постфіксній – після операнда. Значення операнда після застосування обох форм оператора одинакове. Різниця тільки в результаті, який оператор повертає у вираз, в якому він використаний.

Таблиця №2.14. Оператори для інкременту та декременту

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Назва** | **Опис** |
| **++** | Префіксний інкремент: ++var | Збільшує значення змінної на 1 і записує його у цю змінну. Повертає нове значення змінної. |
|  | Постфіксний інкремент: var++ | Збільшує значення змінної на 1 і записує його у цю змінну. Повертає попереднє значення змінної, до інкременту. |
| **--** | Префіксний декремент: --var | Зменшує значення змінної на 1 і записує його у цю змінну. Повертає нове значення змінної. |
|  | Постфіксний декремент: var-- | Зменшує значення змінної на 1 і записує його у цю змінну. Повертає попереднє значення змінної, до декременту. |

Наступний приклад демонструє використання обох форм операторів інкременту та декременту (результат його роботи приведено на рис. 2.7):

int x = 1, y; Console.WriteLine("Початкове значення x: {0}", x); y = x++; Console.WriteLine("y = x++: x: {0}, y: {1}", x, y); x = 1; y = ++x; Console.WriteLine("y = ++x: x: {0}, y: {1}", x, y); x = 1; y = x--; Console.WriteLine("y = x--: x: {0}, y: {1}", x, y); x = 1; y = --x; Console.WriteLine("y = --x: x: {0}, y: {1}", x, y);

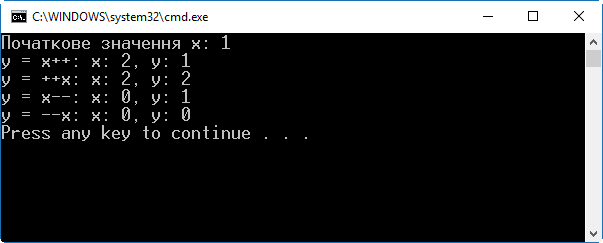


Рис. 2.7. Демонстрація використання обох форм операторів ++ та --