#### Логічні оператори для умов

Логічні операції "І", "АБО", "НЕ" використовують для поєднання умов при порівнянні. Перелік логічних операцій для умов приведено в таблиці 2.15.

Таблиця №2.15. Логічні оператори для поєднання умов

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Назва** | **Опис** |
| **&&** | Логічне "І" | Повертає true, якщо обидва операнди дорівнюють true; повертає false в протилежному випадку |
| **||** | Логічне "АБО" | Повертає true, якщо хоча б один операнд дорівнює true; повертає false в протилежному випадку |
| **!** | Логічне "НЕ" | Повертає true, якщо операнд дорівнює false; повертає false, якщо операнд дорівнює true |

Наприклад, вираз ((*x*> 3)& &(*y*<10)) поверне значення true (істина), якщо значення *x* більше 3 і одночасно (так як використано логічне "І") значення *y* менше 10. В іншому випадку результатом виразу буде false. Таким чином, щоб оператор && повернув true, потрібне виконання умов в обох його операндах.

Вираз ((*x*> 3)|| (*y*<10)) поверне значення true (істина), якщо значення *x* більше 3 або (так як використано логічне "АБО") значення *y* менше 10. Щоб оператор || повернув true, потрібне виконання тільки однієї з умов його операндів.

Вираз (!(*x*> 3)|| (*y*<10)) поверне значення true (істина), якщо значення *x* не більше 3 (використано логічне "НЕ"), або (використано логічне "АБО") значення *y* менше 10.

За допомогою поєднання логічних операторів у програмі можна формувати умови довільної складності.

Наступний фрагмент коду демонструє використання логічних операторів для формування умови з використанням інструкції if (детально її розглянемо далі):

double x, y; y = x = 0;

if (((x > 10) && (y <= 20)) || (x == y)) Console.WriteLine("Ця умова виконується!");

#### Побітові оператори

Інколи при роботі з даними є необхідність побітової їх обробки. З цією метою використовують логічні побітові оператори та оператори побітового зсуву (таблиця 2.16).

Таблиця №2.16. Побітові оператори

|  |  |
| --- | --- |
| **Оператор** | **Назва** |
| **&** | Побітове "І" |
| **|** | Побітове "АБО" |
| **^** | Побітове "Виключаюче АБО" |
| **~** | Побітове "НЕ" |
| **<<** | Зсув бітів вліво |
| **>>** | Зсув бітів вправо |

#### Оператори присвоєння

Оператори присвоєння обчислюють значення виразу справа від оператора і присвоюють це значення змінній зліва від оператора. Ці оператори перелічено у таблиці 2.17.

Таблиця №2.17. Оператори присвоєння

|  |  |
| --- | --- |
| **Оператор** | **Опис** |
| **=** | Базовий простий оператор присвоєння. Результат виразу справа від оператора присвоюється змінній зліва |
|  | **Складні оператори присвоєння** |
| **\*=** | y \*= x еквівалентно до y = y \* x |
| **/=** | y /= x еквівалентно до y = y / x |
| **%=** | y %= x еквівалентно до y = y % x |
| **+=** | y += x еквівалентно до y = y + x |
| **-=** | y -= x еквівалентно до y = y - x |
| **<<=** | y <<= x еквівалентно до y = y << x |
| **>>=** | y >>= x еквівалентно до y = y >> x |
| **&=** | y &= x еквівалентно до y = y & x |
| **^=** | y ^= x еквівалентно до y = y ^ x |
| **|=** | y |= x еквівалентно до y = y | x |

Для складних операторів присвоєння на місці x (див. таблицю 2.17) може бути інший вираз. Наприклад:

Зауважте: дужки

x \*= y – z; // Еквівалентно до x = x \* (y – z) x /= y + z; // Еквівалентно до x = x / (y + z)

Операція присвоєння є виразом, тому повертає результат в те місце програми, де вона записана. Тому результат операції присвоєння також може бути частиною виразу: z = y = x = 5;

Тут спочатку опрацьовується вираз (x=5), в результаті чого змінній x присвоюється значення 5. Далі результат 5 присвоюється змінній y, і в кінці – змінній z.

#### Умовний оператор

Умовний оператор ? : надає зручний та швидкий спосіб вибору одного із двох значень, залежно від результату певної умови. Синтаксис використання цього оператора такий:

Condition ? Expression1 : Expression2

Вираз для умови (Condition) повинен повертати результат логічного типу (bool). Якщо вираз умови дорівнює true, то умовний оператор повертає значення виразу Expression1, а в протилежному випадку – значення виразу Expression2.

Приклад використання цього оператора:

int i = x > y ? 10 : 20;

В результаті виконання цього оператора, якщо значення x більше від y, то змінній i буде присвоєно значення 10. Коли x не більше за y (тобто x менше або дорівнює y), то змінній i буде присвоєно значення 20.