Операции присвоения устанавливают значение. В операциях присвоения участвуют два операнда, причем левый операнд может представлять только модифицируемое именованное выражение, например, переменную

Как и во многих других языках программирования, в C# имеется базовая операция присваивания **=**, которая присвоивает значение правого операнда левому операнду:

|  |  |
| --- | --- |
| 1 | int number = 23; |

Здесь переменной number присваивается число 23. Переменная number представляет левый операнд, которому присваивается значение правого операнда, то есть числа 23.

Также можно выполнять множественно присвоение сразу нескольких переменным одновременно:

|  |  |
| --- | --- |
| 1  2 | int a, b, c;  a = b = c = 34; |

Стоит отметить, что операции присвоения имеют низкий приоритет. И вначале будет вычисляться значение правого операнда и только потом будет идти присвоение этого значения левому операнду. Например:

|  |  |
| --- | --- |
| 1  2 | int a, b, c;  a = b = c = 34 \* 2 / 4; // 17 |

Сначала будет вычисляться выражение 34 \* 2 / 4, затем полученное значение будет присвоено переменным.

Кроме базовой операции присвоения в C# есть еще ряд операций:

* **+=**: присваивание после сложения. Присваивает левому операнду сумму левого и правого операндов: выражение **A += B** равнозначно выражению **A = A + B**
* **-=**: присваивание после вычитания. Присваивает левому операнду разность левого и правого операндов: **A -= B** эквивалентно **A = A - B**
* **\*=**: присваивание после умножения. Присваивает левому операнду произведение левого и правого операндов: **A \*= B** эквивалентно **A = A \* B**
* **/=**: присваивание после деления. Присваивает левому операнду частное левого и правого операндов: **A /= B** эквивалентно **A = A / B**
* **%=**: присваивание после деления по модулю. Присваивает левому операнду остаток от целочисленного деления левого операнда на правый: **A %= B** эквивалентно **A = A % B**
* **&=**: присваивание после поразрядной конъюнкции. Присваивает левому операнду результат поразрядной конъюнкции его битового представления с битовым представлением правого операнда: **A &= B** эквивалентно **A = A & B**
* **|=**: присваивание после поразрядной дизъюнкции. Присваивает левому операнду результат поразрядной дизъюнкции его битового представления с битовым представлением правого операнда: **A |= B** эквивалентно **A = A | B**
* **^=**: присваивание после операции исключающего ИЛИ. Присваивает левому операнду результат операции исключающего ИЛИ его битового представления с битовым представлением правого операнда: **A ^= B** эквивалентно **A = A ^ B**
* **<<=**: присваивание после сдвига разрядов влево. Присваивает левому операнду результат сдвига его битового представления влево на определенное количество разрядов, равное значению правого операнда: **A <<= B** эквивалентно **A = A << B**
* **>>=**: присваивание после сдвига разрядов вправо. Присваивает левому операнду результат сдвига его битового представления вправо на определенное количество разрядов, равное значению правого операнда: **A >>= B** эквивалентно **A = A >> B**

Применение операций присвоения:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | int a = 10;  a += 10; // 20  a -= 4; // 16  a \*= 2; // 32  a /= 8; // 4  a <<= 4; // 64  a >>= 2; // 16 |

Операции присвоения являются правоассоциативными, то есть выполняются справа налево. Например:

|  |  |
| --- | --- |
| 1  2  3 | int a = 8;  int b = 6;  int c = a += b -= 5; // 9 |

В данном случае выполнение выражения будет идти следующим образом:

1. b -= 5 (6-5=1)
2. a += (b-=5) (8+1 = 9)
3. c = (a += (b-=5)) (c = 9)