

В C# предусмотрены также специальные, ***укороченные****,* варианты логических операторов

И и ИЛИ, предназначенные для получения более эффективного кода. Поясним

это на следующих примерах логических операций. Если первый операнд логической

операции И имеет ложное значение (false), то ее результат будет иметь ложное значение

независимо от значения второго операнда. Если же первый операнд логической

операции ИЛИ имеет истинное значение (true), то ее результат будет иметь истинное

значение независимо от значения второго операнда. Благодаря тому, что значение второго

операнда в этих операциях вычислять не нужно, экономится время и повышается

эффективность кода.

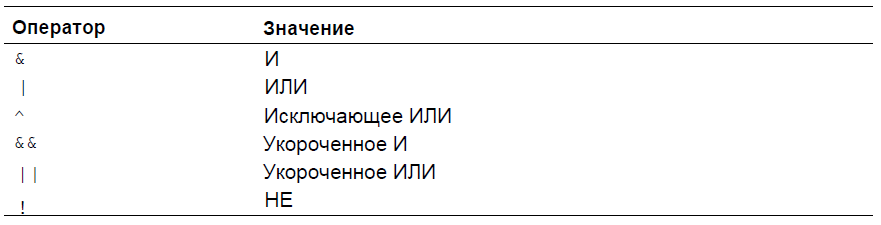
Укороченная логическая операция И выполняется с помощью оператора &&, а укороченная

логическая операция ИЛИ — с помощью оператора ||. Этим укороченным

логическим операторам соответствуют обычные логические операторы & и |. Единственное

отличие укороченного логического оператора от обычного заключается в том,

что второй его операнд вычисляется только по мере необходимости.

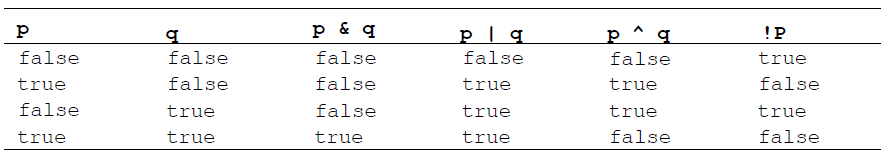


Операнды логических операторов должны относиться к типу bool, а результат выполнения

логической операции также относится к типу bool. Логические операторы

&, |, ^ и ! поддерживают основные логические операции И, ИЛИ, исключающее ИЛИ

и НЕ в соответствии с приведенной ниже таблицей истинности.



Как следует из приведенной выше таблицы, результатом выполнения логической

операции исключающее ИЛИ будет истинное значение (true), если один и только

один ее операнд имеет значение true.

Эти логические операции могут быть построены с помощью логических

операторов, поддерживаемых в С#. Следовательно, в C# предусмотрен такой набор

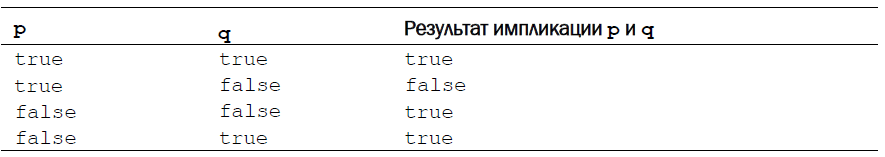
логических операторов, которого достаточно для построения практически любой

логической операции, в том числе импликации. *Импликация* — это двоичная операция,

результатом которой является ложное значение только в том случае, если левый

ее операнд имеет истинное значение, а правый — ложное. (Операция импликации отражает

следующий принцип: истина не может подразумевать ложь.)



**Составные операторы присваивания**

Составные операторы присваивания записываются более кратко, чем их несоставные

эквиваленты. Поэтому их иногда еще называют *укороченными операторами*

*присваивания.*

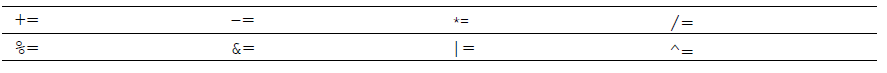
У составных операторов присваивания имеются два главных преимущества. Во-

первых, они более компактны, чем их "несокращенные" эквиваленты. И во-вторых, они

дают более эффективный исполняемый код, поскольку левый операнд этих операторов

вычисляется только один раз. Именно по этим причинам составные операторы присваивания

чаще всего применяются в программах, профессионально написанных на С#.



**Поразрядные операторы**

В C# предусмотрен ряд *поразрядных* операторов, расширяющих круг задач, для решения

которых можно применять С#. Поразрядные операторы воздействуют на отдельные

двоичные разряды (биты) своих операндов. Они определены только для целочисленных

операндов, поэтому их нельзя применять к данным типа bool, float или

double.

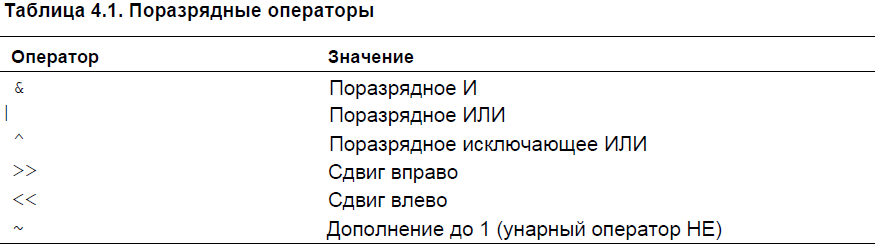
Эти операторы называются *поразрядными,* поскольку они служат для проверки,

установки или сдвига двоичных разрядов, составляющих целое значение. Среди прочего

поразрядные операторы применяются для решения самых разных задач программирования

на уровне системы, включая, например, анализ информации состояния

устройства. Все доступные в C# поразрядные операторы приведены в табл. 4.1.



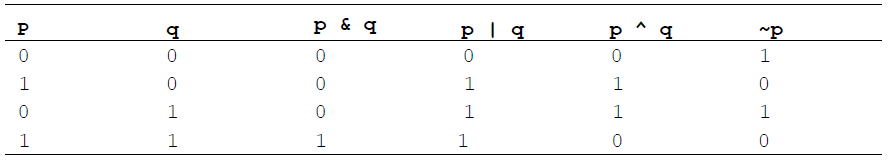
Поразрядные операторы И, ИЛИ, исключающее ИЛИ и НЕ обозначаются следующим

образом: &, |, ^ и ~. Они выполняют те же функции, что и их логические аналоги,

рассмотренные выше. Но в отличие от логических операторов, поразрядные операторы

действуют на уровне отдельных двоичных разрядов. Ниже приведены результаты

поразрядных операций с двоичными единицами и нулями.

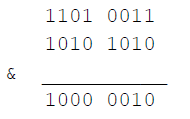


С точки зрения наиболее распространенного применения поразрядную операцию

И можно рассматривать как способ подавления отдельных двоичных разрядов. Это

означает, что если какой-нибудь бит в любом из операндов равен 0, то соответствующий

бит результата будет сброшен в 0. Например:



**Оператор ?**

Оператор ? относится к числу самых примечательных в С#. Он представляет собой

условный оператор и часто используется вместо определенных видов конструкций

if-then-else. Оператор ? иногда еще называют *тернарным,* поскольку для него требуются

три операнда. Ниже приведена общая форма этого оператора.

***Выражение1 ? Выражение2 : Выражение3;***

В общем тут все понятно, как и в С++.

**Предшествование операторов**

Тут все просто, в таблице ниже, перечислен приоритет операторов, которые выполняются в порядке их отображения. То есть, список приоритетности, который исполняет компилятор.

