^ 运算符（C# 参考）^ Operator (C# Reference)

* ‎2015‎年‎07‎月‎20‎日
* 作者
  + [Bill Wagner](https://github.com/BillWagner)
  + [](https://github.com/olprod)
  + [](https://github.com/OpenLocalizationService)
  + [yishengjin1413](https://github.com/yishengjin1413)
  + [Saisang Cai](https://github.com/Saisang)

本文内容

1. [备注Remarks](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/operators/xor-operator#remarks)
2. [示例Example](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/operators/xor-operator#example)
3. [另请参阅See Also](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/operators/xor-operator#see-also)

针对整型类型和 bool 预定义了二元 ^ 运算符。Binary ^ operators are predefined for the integral types and bool. 对于整型类型，^ 会计算其操作数的按位异或。For integral types, ^ computes the bitwise exclusive-OR of its operands. 对于 bool 操作数，^ 计算其操作数的逻辑异或；即，当且仅当其一个操作数为 true 时，结果才为 true。For bool operands, ^ computes the logical exclusive-or of its operands; that is, the result is true if and only if exactly one of its operands is true.

备注Remarks

用户定义的类型可以重载 ^ 运算符（请参阅[运算符](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/keywords/operator)）。User-defined types can overload the ^ operator (see [operator](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/keywords/operator)). 对整数类型的操作通常可用于枚举。Operations on integral types are generally allowed on enumeration.

示例Example

C# 复制

class XOR

{

static void Main()

{

// Logical exclusive-OR

// When one operand is true and the other is false, exclusive-OR

// returns True.

Console.WriteLine(true ^ false);

// When both operands are false, exclusive-OR returns False.

Console.WriteLine(false ^ false);

// When both operands are true, exclusive-OR returns False.

Console.WriteLine(true ^ true);

// Bitwise exclusive-OR

// Bitwise exclusive-OR of 0 and 1 returns 1.

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0x0 ^ 0x1, 2));

// Bitwise exclusive-OR of 0 and 0 returns 0.

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0x0 ^ 0x0, 2));

// Bitwise exclusive-OR of 1 and 1 returns 0.

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0x1 ^ 0x1, 2));

// With more than one digit, perform the exclusive-OR column by column.

// 10

// 11

// --

// 01

// Bitwise exclusive-OR of 10 (2) and 11 (3) returns 01 (1).

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0x2 ^ 0x3, 2));

// Bitwise exclusive-OR of 101 (5) and 011 (3) returns 110 (6).

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0x5 ^ 0x3, 2));

// Bitwise exclusive-OR of 1111 (decimal 15, hexadecimal F) and 0101 (5)

// returns 1010 (decimal 10, hexadecimal A).

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0xf ^ 0x5, 2));

// Finally, bitwise exclusive-OR of 11111000 (decimal 248, hexadecimal F8)

// and 00111111 (decimal 63, hexadecimal 3F) returns 11000111, which is

// 199 in decimal, C7 in hexadecimal.

Console.WriteLine("Bitwise result: {0}", Convert.ToString(0xf8 ^ 0x3f, 2));

}

}

/\*

Output:

True

False

False

Bitwise result: 1

Bitwise result: 0

Bitwise result: 0

Bitwise result: 1

Bitwise result: 110

Bitwise result: 1010

Bitwise result: 11000111

\*/

上一示例中的 0xf8 ^ 0x3f 计算执行了以下两个二进制值的按位异或，这与十六进制值 F8 和 3F 对应：The computation of 0xf8 ^ 0x3f in the previous example performs a bitwise exclusive-OR of the following two binary values, which correspond to the hexadecimal values F8 and 3F:

1111 1000

0011 1111

异或的结果是 1100 0111，即十六进制中的 C7。The result of the exclusive-OR is 1100 0111, which is C7 in hexadecimal.