Операторы со знаком вопроса: ??, ?. и ?:

admin
December 2021 edited March 5

Null-coalescing

?? - это Null-coalescing оператор. Он возвращает значение левого операнда, если оно не равно нулю. В противном случае возвращается значение правого.

```
string str=null, str2="test";
WriteLine(str ?? str2); //test
```

?? - настоящий оператор, а операторы ?. и ?: являются синтаксическим сахаром для функций `IfNotNull(объект, значение)` и `If(условие, значение_1, значение_2)` .

Ternary

?: - условный (тернарный) оператор: `условие ? true-выражение : false-выражение` .

```
WriteLine(5>4 ? "hello" : "world"); //hello
WriteLine(5<4 ? "hello" : "world"); //world
//Теперь посмотрим, что видит интерпретатор. Для этого создадим функцией Expr() выражение.
WriteLine(Expr(5<4 ? "hello" : "world")); //@If(5<4,"hello","world")
```

Т.е. `5<4 ? "hello" : "world"` - то же самое, что `@If(5<4,"hello","world")` . Символ @ перед именем функции указывает интерпретатору, что нужно вызвать базовую (встроенную) функцию, поэтому перегрузить операторы ?. и ?: не получится.
Этот оператор можно использовать без третьего аргумента.

```
WriteLine(5>4? 1); //1
WriteLine(5<4? 1); //0
```

```
WriteLine(Expr(5<4? 1)); //@If(5<4,1,)

WriteLine(5>4? Now()); //24.12.2021 12:44:23
WriteLine(5<4? Now()); //01.01.0001 0:00:00
WriteLine(Expr(5<4? Now())); //@If(5<4,Now(),)
```

Если условие не выполняется, то возвращается значение по умолчанию для типа true-выражения.

```
//А ещё можно так использовать:
bool b=true;
b ? WriteLine("TRUE!") : WriteLine("FALSE!"); //TRUE!
b ? WriteLine("OK!"); //OK!
```

Optional chaining

Оператор **?.** заменяется на функцию IfNotNull, которая возвращает значение второго аргумента, если первый не равен null, а если равен, то возвращает null. Это оператор опциональной последовательности (Optional chaining operator). Если в цепочке объектов какой-то элемент равен null, то возвращается null.

```
Foo foo=new Foo("Hello");
WriteLine("Str: "+foo?.Str); //Str: Hello
foo=null;
WriteLine("Str: "+(foo?.Str==null)); //Str: True

WriteLine(Expr(foo?.Str)); //@IfNotNull((var °chainItem_3=foo),_(°chainItem_3.Str,°chainItem_3=null))

//Кстати, вместо ?. можно делать так:
WriteLine("Str: "+_(foo.Str)("error_value")); //Str: error_value
//_(foo.Str) - это функция возвращающая значение первого аргумента, а ("error_value") - какое значение вернуть в случае, если функция перед в

class Foo{
    public string Str;
    New(string s){
        Str=s;
```

```
    }
}
```

°chainItem_3 - это автоматически созданная переменная для хранения значения текущего звена, чтобы избежать ситуации, при которой значение проверенное в условии может быть изменено другим потоком во время работы функции IfNotNull до получения значения второго аргумента. К тому же, это кеширование значений позволяет не вычислять по несколько раз одни и те же сложные выражения. Имена таких служебных переменных начинаются со знака градуса, который нельзя использовать для пользовательских переменных.
Из примера видно, что вторым аргументом IfNotNull идёт: `_(°chainItem_3 .Str,°chainItem_3=null)`. Функция `_(аргумент_1, аргумент_2, аргумент_3, ...)` возвращает значение первого аргумента и высчитывает остальные. Т.е. будет возвращено значение `°chainItem_3.Str` и выполнено `°chainItem_3=null`, чтобы обнулить временную переменную.

А вот, как для элементов массива использовать:

```
string[] arr={"foo", "bar", "baz"};
WriteLine(arr?[2]); //baz
arr=null;
WriteLine(arr?[2]==null); //True
//WriteLine(arr[2]); //выдаст исключение NullReferenceException
```

Есть оператор **?->** для Reflection-цепочек:

```
WriteLine("hello world"->Substring(6)?->ToUpper()); //WORLD
```

В выражениях для констант optional chaining не поддерживается, т.к. в них нельзя использовать переменные, и chainItem-переменная вызовет ошибку.

```
const string t="test";
const string s=t?.ToUpper(); //выдаст ошибку "Variable '°chainItem_1' not found."
```

Оператора ??= нет.