Передача параметров в метод по ссылке. Операторы ref и out

В C# значения переменных по-умолчанию передаются по значению (в метод передается локальная копия параметра, который используется при вызове). Это означает, что мы не можем внутри метода изменить параметр из вне:

```
public static void ChangeValue(object a)
{
   a = 2;
}

static void Main(string[] args)
{
   int a = 1;
   ChangeValue(a);
   Console.WriteLine(a); // 1
   Console.ReadLine();
}
```

Чтобы передавать параметры по ссылке, и иметь возможность влиять на внешнюю переменную, используются ключевые слова ref и out.

Ключевое слово ref

Чтобы использовать ref, это ключевое слово стоит указать перед типом параметра в методе, и перед параметром при вызове метода:

```
public static void ChangeValue(ref int a)
{
   a = 2;
}
static void Main(string[] args)
{
   int a = 1;
   ChangeValue(ref a);
   Console.WriteLine(a); // 2
   Console.ReadLine();
}
```

В этом примере мы изменили значение внешней переменной внутри метода.

Особенностью ref является то, что переменная, которую мы передаем в метод, обязательно должна быть проинициализирована значением, иначе компилятор выдаст ошибку «Use of unassigned local variable 'a'». Это является главным отличием ref от out.

Ключевое слово out

Out используется точно таким же образом как и ref, за исключением того, что параметр не обязан быть проинициализирован перед передачей, но при этом в методе переданному параметру обязательно должно быть присвоено новое значение:

```
public static void ChangeValue(out int a)
{
   a = 2;
}
static void Main(string[] args)
{
   int a;
   ChangeValue(out a);
   Console.WriteLine(a); // 2
   Console.ReadLine();
}
```

Если не присвоить новое значение параметру out, мы получим ошибку «The out parameter 'a' must be assigned to before control leaves the current method»

Производительность

Учитывая тот факт, что по умолчанию в метод передаются параметры по значению и создаются их копии в стеке, при использовании сложных типов данных (пользовательские структуры), или если метод вызывается много раз, это плохо скажется на производительности. В таком случае также стоит использовать ключевые слова ref и out.

Если говорить в целом о ссылочных типах и типах значений, то производительность приложения упадет, если использовать только ссылочные типы. На создание переменной ссылочного типа в куче выделяется память под данные, а в стеке под ссылку на эти данные. Для типов значений память выделяется только в стеке. Время на размещение данных в стеке меньше, чем в куче, это также идет в плюс типам значений в плане производительности.