Global symbol requires explicit package name

strict (/search/strict) my (/search/my) package (/search/package) global symbol (/search/global symbol)

Global symbol requires explicit package name это типичная, и, на мой взгляд, весьма вводящее в заблуждение сообщение об ошибке в Perl. Во всяком случае, для новичков.

Простой перевод - "Надо объявить переменную с помощью **my**."

Простейший пример

```
1. use strict;
2. use warnings;
3.
4. $x = 42;
```

И ошибка

```
Global symbol "$x" requires explicit package name at ...
(Глобальный символ "$x" требует явного имени модуля на строке...)
```

Хотя само по себе сообщение об ошибке корректно, от этого мало толку для новичка в Perl-программировании. Он, возможно, даже не знает, что такое модули. И также ему непонятно, что может быть более явно, чем $x.

Это ошибка генерируется из-за **use strict**.

Вот объяснение из документации:

Это выражение генерирует ошибку компиляции, если вы обращаетесь к переменной, которая не была объявлена с помощью "our" или "use vars", локально объвлена с помощью "my()", или не была полностью определена.

Стоит надеяться, что новичок будет начинает каждый свой скрипт с **use strict**, и наверняка узнает о **my** гораздо раньше, чем о других вариантах.

Я не знаю, возможно и следует ли изменить этот текст в perl'е. Цель этой статьи не в этом. Цель в том, чтобы помочь новичкам понять для себя, что значит это сообщение об ошибке.

Чтобы избавиться от этого сообщение об ошибке, нужно написать:

```
1. use strict;
2. use warnings;
3.
4. my $x = 42;
```

То есть, нужно **объявить переменную с помощью my, прежде чем использовать ее в первый раз**. That is, one needs to **declare the variable using my before its first use.**.

Плохое решение

Другое "решение" - убрать **strict**:

```
1. #use strict;
2. use warnings;
3.
4. $x = 23;
```

это сработает, но выдаст предупреждение Name "main::x" used only once: possible typo at ... (/name-used-only-once-possible-typo).

Так или иначе, в нормальной ситуации вы не станете вести машину, не пристегнувшись, не так ли?

Пример 2: область видимости

Другой случай, часто наблюдаемый у новичков:

```
1. use strict;
2. use warnings;
3.
4. my $x = 1;
5.
6. if ($x) {
7. my $y = 2;
8. }
9.
10. print $y;
```

Здесь мы получим ту же ошибку, что и в предыдущем примере:

```
Global symbol "$y" requires explicit package name at ...
```

что для многих удивительно. Особенно, когда они начинают программировать. В конце концов, они же объявили $y с помощью my .

Для начала, здесь есть небольшая визуальная проблема. Нет отступа перед `my $y = 2;` . Если бы мы сделали отступ в пару пробелов или табуляцию, как в следующем примере, источник проблемы мог бы стать более очевидным:

```
1. use strict;
2. use warnings;
3.
4. my $x = 1;
5.
6. if ($x) {
7.     my $y = 2;
8. }
9.
10. print $y;
```

Проблема в том, что переменная `$y` объявлена внутри блока(между фигурными скобками), а это значит, что вне пределов блока она не существует. Это называется **областью видимости** переменной (/oblast-vidimosti-v-perl).

Сама идея **областей видимости** отличается в разных языках программирования. В Perl блок, ограниченный фигурными скобками, создает область видимости. То, что объявлено внутри нее с помощью `my` , не будет доступно за границами блока.

(К слову, выражение `$x = 1` здесь добавлено только чтобы создать правдоподобное условие для создания блока. Другими словами, условие `if ($x) {` нужно здесь, чтобы сделать пример более реальным.)

Решение здесь - либо вызвать `print` внутри блока:

```
1. use strict;
2. use warnings;
3.
4. my $x = 1;
5.
6. if ($x) {
7.     my $y = 2;
8.     print $y;
9. }
```

либо объявить переменную за границами блока (а не внутри!):

```
 1. use strict;
 2. use warnings;
 3.
 4. my $x = 1;
 5. my $y;
 6.
 7. if ($x) {
 8.     $y = 2;
 9. }
10.
11. print $y;
```

Какой способ выбрать, зависит уже от конкретной задачи. Это лишь синтаксически допустимые решения.

Конечно, если мы забудем убрать `my` из блока, или если `$x` будет FALSE, то мы получим предупреждение Use of uninitialized value (/use-of-uninitialized-value).

Другие способы

Объяснение того, что делают выражения `our` и `use vars` , или как можно полностью определить имя переменной, мы оставим для другой статьи.

Переводчик
Vladimir Zatoloka
(https://www.linkedin.com/)

Автор
Gabor Szabo
(https://www.linkedin.com/in/szabgab/)

Published on 2013-07-09

If you have any comments or questions, feel free to post them on the source of this page in GitHub. Source on GitHub. (https://github.com/szabgab/perlmaven.com/tree/main/sites/ru/pages//global-symbol-requires-explicit-package-name.txt) Comment on this post (https://github.com/szabgab/perlmaven.com/issues/new?title=&body=https://ru.perlmaven.com/global-symbol-requires-explicit-package-name)

Bahasa Indonesia (https://id.perlmaven.com/simbol-global)
Deutsch (https://de.perlmaven.com/global-symbol-requires-explicit-package-name)
English (https://perlmaven.com/global-symbol-requires-explicit-package-name)
Italiano (https://it.perlmaven.com/global-symbol-requires-explicit-package-name)
Português (https://br.perlmaven.com/simbolo-global-requer-nome-de-pacote-explicito)

Română (https://ro.perlmaven.com/simbolul-global-necesita-nume-de-pachet-explicit)

Русский (https://ru.perlmaven.com/global-symbol-requires-explicit-package-name)

✔

体中文 (https://cn.perlmaven.com/global-symbol-requires-explicit-package-name)

(https://ko.perlmaven.com/global-symbol-requires-explicit-package-name)

about the translations (https://perlmaven.com/about#translations)

Name "main::x" used only once: possible typo at ... (/name-used-only-once-possible-typo)

Use of uninitialized value (/use-of-uninitialized-value)