Министерство образования Республики Беларусь

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

**Язык программирования “PERL”**

|  |  |
| --- | --- |
| Выполнил: | студенты группы 772302  Углянец А.В., Нагула А.А. |
| Проверил: | Болтак С.В. |

Минск 2018

**ВВЕДЕНИЕ**

**Perl** – это интерпретируемый язык, приспособленный для обработки произвольных текстовых файлов, извлечения из них необходимой информации и выдачи сообщений. Он также удобен для написания различных системных программ. Этот язык прост в использовании, эффективен, но про него трудно сказать, что он элегантен и компактен.

Язык можно охарактеризовать скорее как, чем красивый. Главными достоинствами языка являются поддержка различных [парадигм](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) ([процедурный](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [функциональный](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) стили программирования), контроль за памятью (без сборщика мусора, основанного на циклах), встроенная поддержка обработки текста, а также большая коллекция [модулей](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) сторонних разработчиков.

Название языка официально расшифровывается как Practical Extraction and Report Language («практический язык для извлечения данных и составления отчётов»), а в шутку — как Pathologically Eclectic Rubbish Lister («патологически эклектичный перечислитель мусора»). Символом языка Perl является [верблюд](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D0%B1%D0%BB%D1%8E%D0%B4) — не слишком красивое, но очень выносливое животное, способное выполнять тяжёлую работу.

Основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с [регулярными выражениями](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F), встроенная в синтаксис. Перл унаследовал много свойств от языков [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [AWK](https://ru.wikipedia.org/wiki/AWK), скриптовых языков [командных оболочек UNIX](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BE%D0%BB%D0%BE%D1%87%D0%BA%D0%B0_UNIX).

### **ИСТОРИЯ**

Perl был создан [Ларри Уоллом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BE%D0%BB%D0%BB,_%D0%9B%D0%B0%D1%80%D1%80%D0%B8), [лингвистом](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B3%D0%B2%D0%B8%D1%81%D1%82) по образованию, в 1986 году как инструмент для администрирования и конфигурирования системных ресурсов сети, состоящей из Unix-компьютеров.

Причина возникновения. Как утверждал Ларри, помогла ему создать новый язык лень. Но именно та лень, которая позволила решить важную задачу: как избежать составления многочисленного числа программ на разных языках, которые были неотъемлемой частью инструментальных средств UNIX. Слишком утомительной была такая процедура. Достоинство нового языка оценили другие, так как в нем гармонично сочетались возможности обработки файлов и системного администрирования. А это именно два необходимых факта, которые нужны во время программирования системы UNIX. Следует отметить, что у языка программирования Perl есть небольшое отличие от других подобных языков: он появился оттого, что в нем назрела необходимость, а не для того, чтобы поразить пользователей или придумать очередное красивое средство, которое не сыграет особой роли.

Новый язык программирования Perl открыл очень богатые возможности в тех случаях, когда требуется обработать текст, устойчивые выражения. Отличает Perl большие варианты вспомогательных модулей. Со временем этот уникальный язык расширил свои границы, и теперь широко используется в веб-разработках и играх, если необходимо сетевое программирование или графическая разработка интерфейсов для пользователей. Язык прижился и полюбился за легкость использования.

Ларри многое позаимствовал из других языков программирования. Например: командные оболочки UNIX. Переменные маркированы ведущими знаками, они четко выражают тип переменной и помогают этим переменным являться в строках интерполированными. Встроенные функции обеспечивают инструментарий, а он используется для программирования оболочки. Но с собственными задачами новый язык справлялся гораздо быстрее, и его, поэтому начали применять не только для решения задач системного администрирования.

### ОСОБЕННОСТИ ЯЗЫКА

Общая структура Perl в общих чертах ведёт своё начало от языка [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)). Perl — [процедурный](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) по своей природе, имеет [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), выражения [присваивания](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), блоки кода, отделяемые [фигурными скобками](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D0%B1%D0%BA%D0%B8), управляющие структуры и [функции](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).

Perl также заимствует ряд свойств из языков программирования [командных оболочек UNIX](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BE%D0%BB%D0%BE%D1%87%D0%BA%D0%B0_UNIX). Все переменные маркируются ведущими знаками, которые точно выражают тип данных переменной в этом контексте (например, скаляр, массив, хеш). Важно, что эти знаки позволяют переменным быть интерполированным в строках. Perl обладает множеством встроенных функций, которые обеспечивают инструментарий, часто используемый для программирования оболочки, например сортировку или вызов системных служб.

Perl заимствует [массивы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) из [Лиспа](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%81%D0%BF), [регулярные выражения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F) из [AWK](https://ru.wikipedia.org/wiki/AWK) и [sed](https://ru.wikipedia.org/wiki/Sed" \o "Sed), из AWK также позаимствованы [хеши](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0" \o "Хеш-таблица) («ассоциативные массивы»). Регулярные выражения облегчают выполнение многих задач по парсингу, обработке текста и манипуляций с данными.

Perl 5 добавил поддержку сложных типов данных, [первоклассных функций](http://en.wikipedia.org/wiki/First-class_function) и объектную модель. В последнюю входят [ссылки](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), пакеты, выполнение методов от класса, переменные с лексическим объявлением [области видимости](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C_%D0%B2%D0%B8%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8), а также [директивы](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%80%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) компилятора. Главнейшим усовершенствованием, представленным в Perl 5, стала возможность помещать код в «пакеты» (*package*) в качестве модулей для повторного использования. Ларри Уолл позже заметил, что «Весь замысел модульной системы Perl 5 сводился к поощрению роста культуры Perl, а не строчек кода»[[](https://ru.wikipedia.org/wiki/Perl#cite_note-7).

Все версии Perl выполняют автоматическую типизацию данных и автоматический контроль над памятью. Интерпретатор знает тип и запросы памяти каждого объекта программы, он распределяет и освобождает память, производя [подсчёт ссылок](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D1%87%D1%91%D1%82_%D1%81%D1%81%D1%8B%D0%BB%D0%BE%D0%BA). Перевод одного типа данных в другой — например, числа в строку — происходит автоматически во время исполнения, невозможные для выполнения переводы типов данных приводят к фатальной ошибке.

Но все разработки роднит несколько факторов: выполняется автоматическая типизация данных, проводится автоматический контроль над памятью. Благодаря интерпретатору известны не только типы, но и запросы памяти каждого объекта программы, и ему не составляет труда распределить память, освободить при подсчете ссылок. Нужно отметить, что Perl-программирование отличает интерпретатор от других. Доказательством тому являются трансляция программы в промежуточный байт-код, а уже потом начинает выполняться. В сравнение: традиционный работает немного иначе – вводимый оператор интерпретируется и мгновенно выполняется, что чревато синтаксическими ошибками во время рабочего процесса. Perl позволяет обнаружить такого рода ошибки уже в период трансляции в бай-код.

Для того чтобы перевести один вид данных в другой, потребуется автоматический процесс, если же невозможны переводы каких-либо видов данных, это может привести к роковой ошибке. Заслуга версии 5 004 в том, что здесь имеется пакет UNIVERSAL, который языку дал основной объект, а от него произошли все классы – на автоматической основе. Появилась возможность запросить версию модулей. Стало реальностью поддерживать запрашивать версию модулей. В дополнение к этому perl стал поддерживать многие операционные системы, и Microsoft Windows в том числе. Версия 5.005, вышедшая в 1998 году, значительно повысила качество процессора регулярных выражений, улучшила новые перехваты в бэкенд, что решилось с помощью определенных модулей и расширила поддержку операционных систем.

**СИНТАКСИС ЯЗЫКА**

Синтаксис Perl имеет много общего с синтаксисом языков [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [AWK](https://ru.wikipedia.org/wiki/AWK), [sed](https://ru.wikipedia.org/wiki/Sed" \o "Sed) и [Bourne shell](https://ru.wikipedia.org/wiki/Bourne_shell" \o "Bourne shell).

Первая строка исходного кода может начинаться с «#!/Путь/к/Perl [-ключи]» — что указывает системе путь к интерпретатору Perl для выполнения программы в UNIX-системах и выполнения их на [веб-сервере](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80).

Простейшая программа «[Hello, world!](https://ru.wikipedia.org/wiki/Hello,_world!" \o "Hello, world!)» выглядит следующим образом:

#!/usr/bin/perl

print "Hello, world!\n";

либо для [CGI](https://ru.wikipedia.org/wiki/CGI):

#!/usr/bin/perl

print "Content-type: text/html\n\n";

print "Hello, world!";

либо используя один из post modern web-framework

use Mojolicious::Lite;

get '/' => {text => 'Hello World!'};

app->start;

либо используя [PSGI](https://ru.wikipedia.org/wiki/PSGI) спецификацию

my $app = sub {

return [200, ['Content-Type' => 'text/plain'], ["hello, world\n"]];

}

### Типы данных

Основные типы данных: [скаляр](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BB%D1%8F%D1%80), [массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [хеш-таблица](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0), [функция](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [указание на файл](http://en.wikipedia.org/wiki/Filehandle), [запись таблицы символов](https://ru.wikipedia.org/wiki/Perl#Typeglob_values). Переменные разных типов отличаются знаком, который стоит перед именем переменной:

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Символ** | **Пример** | **Описание** |
| [Скаляр](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BB%D1%8F%D1%80) | $ | $foo | единичное значение; может быть числом, [строкой](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF), указанием на файл, а также [ссылкой](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). |
| [Массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) | @ | @foo | Упорядоченная коллекция скаляров. |
| [Хеш-таблица](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0) | % | %foo | Коллекция из пар строка-скаляр; строки называются *ключами*, а скаляры — *значениями*. Синоним — [ассоциативный массив](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2). |
| [Файловый дескриптор](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B4%D0%B5%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%80) | < > | $foo или FOO | Представление открытого файла или другой цели, открытой для чтения и/или записи. |
| [Функция](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) | & | &foo | Отрезок кода, который принимает аргументы, исполняет действия и возвращает значения. |
| [Запись таблицы символов](https://ru.wikipedia.org/wiki/Perl#Typeglob_values) | \* | \*foo | Запись таблицы символов для всех типов с именем 'foo'. |

#### Скаляр**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=11)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=11)**]**

Скалярные переменные используются для хранения одиночных значений. Они могут содержать числа, строки и [ссылки](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) на другие [объекты](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Перед именем скалярной переменной необходимо ставить знак доллара '$'. Тип скалярной переменной не фиксирован (в отличие от, например, языка [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F))) и определяется динамически в зависимости от контекста. Например,

$x = 10; # число

$y = $x + 1; # используем $x как число…

$z = $x . 'string'; # … а теперь — как строку

$ref = \$x; # $ref является указателем на $x

$$ref = 10; # $x содержит значение 10Строковые данные должны быть помещены в кавычки. Perl использует два вида кавычек — одиночные и двойные. Помещение строки в двойные кавычки позволяет автоматически заменить встречающиеся имена переменных на их значения. Заключение строк в одиночные кавычки предотвращает такое поведение:

## Пример интерполяции переменных в двойных кавычках

$pi = 3.141592654;

$var1 = 'Значение Pi - $pi\n';

print $var1; # Значение Pi - $pi\n

$var2 = "Значение Pi - $pi\n";

print $var2; # Значение Pi - 3.141592654Для включения двойных кавычек в строку, которая была помещена в двойные кавычки, следует использовать \". Для включения одинарных кавычек в строку из одинарных кавычек — \'. Для удобства работы со строками, содержащими много знаков кавычек внутри, Perl позволяет использовать альтернативные способы закавычивания. Альтернативная форма использует любую пару знаков, помещённых после буквы q (имитирует поведение одинарных кавычек) или qq (имитирует поведение двойных кавычек):

## Пример использования альтернативных кавычек

$pi = 3.141592654;

$link1 = q{<a href="http://ru.wikipedia.org/wiki/Pi">$pi</a>};

print $link1; # <a href="http://ru.wikipedia.org/wiki/Pi">$pi</a>

$link2 = qq{<a href="http://ru.wikipedia.org/wiki/Pi">$pi</a>};

print $link2; # <a href="http://ru.wikipedia.org/wiki/Pi">3.141592654</a>Ещё один тип кавычек в Perl — обратные кавычки. В них помещаются программы операционной системы, вывод которых может быть передан интерпретатору Perl. Выполнение программ всегда заканчиваются символом новой строки — \n:

#### ## Пример использования обратных кавычек для исполнения команд UNIX

#### $space = `du -s -k /usr/local/bin`;

#### print $space; # 6264 /usr/local/bin

#### $sysuptime = `uptime`;

#### print $sysuptime; # 2:24 up 1 day, 9:05, 1 user, load averages: 0.26 0.32 0.33Массив**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=12)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=12)**]**

[Массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) является упорядоченным списком [скаляров](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%B0%D0%BB%D1%8F%D1%80). Каждый элемент массива имеет порядковый индекс, с помощью которого к нему можно получить доступ. Нумерация элементов начинается с нуля, то есть первый элемент списка имеет индекс 0. Перед именем переменной-массива необходимо ставить знак '@', а для доступа к определённому элементу массива использовать знак '$', так как отдельный элемент массива является скаляром. Многомерные массивы можно смоделировать, помещая в список ссылки на другие списки.

#### @array = (1,2,3,4); # записали в массив @array четыре элемента

#### print $array[2]; # напечатать третий элемент

#### print @array[2,3]; # напечатать третий и четвёртый элемент.

#### # Здесь используется @, так как результат операции среза массива тоже является массивом.Хеш-таблица**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=13)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=13)**]**

* Хеш-таблица представляет собой [ассоциативный массив](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), позволяющий ассоциировать строку (называемую «ключом») со скаляром (значение). Строка называется ключом, а скаляр в хеш-таблице значением. Перед именем переменной-списка необходимо ставить знак процента '%', а для доступа к определенному элементу массива необходимо ставить знак '$'.

Хеш-таблицу можно инициализировать массивом или списком, состоящим из неограниченного числа последовательностей (ключ, значение).

#### %hash = (

#### cat => 'kitten', # здесь => - это так называемая "длинная запятая", в этом примере

#### dog => 'puppy', # по функции полностью аналогичная обыкновенной "," за исключением

#### cow => 'calf' # того, что литералы слева неявно заключаются в кавычки

#### );

#### print $hash{'cat'}; #Напечатает kitten

#### print join("-", keys %hash) ; # Вывод всех ключей. Напечатает cat-dog-cow

#### print join("-", values %hash) ; # Вывод всех значений. Напечатает kitten-puppy-calf

#### print join("-", %hash) ; # Напечатает cat-kitten-cow-calf-dog-puppy, так как в списковом контексте хэш автоматически

#### # разворачивается в список (ключ, значение). Так как ключи в хэше не сортируются,

#### # пары каждый раз будут выводиться в произвольном порядке.Функция**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=14)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=14)**]**

* Функция представляет собой фрагмент исполняемого кода. Функция всегда возвращает какое-либо значение. Если возвращаемое значение явно не указано оператором return, возвращается последнее вычисленное значение. Если в функции используется return без параметров, то в зависимости от контекста вызова функции возвращается неопределённое значение undef, пустой список или вообще ничего.
* Константа представляет собой неизменяемое значение. Константа не является встроенным типом языка и эмулируется при помощи функций. Предпочтительно задавать константы с помощью стандартного модуля constant, чтобы в будущем не зависеть от возможных изменений в реализации констант. Использование:

#### use constant MY => 2;

#### print MY;Запись таблицы символов**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=16)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=16)**]**

* Запись таблицы символов (typeglob) — специальный тип, в котором хранятся ссылки на все переменные других типов с таким названием, а также на символы, которые часто использует для хранения файловый дескриптор, представляющих собой указатель на файл, устройство или PIPE канал, открытые для записи, чтения или для записи и чтения.

Использование:

$s = readline \*STDIN; #Читаем одну строку из дескриптора STDIN (стандартный поток ввода); скалярный контекст.

@values = <FILE>; #Читаем все строчки из некоторого открытого и связанного с дескриптором FILE файла; списковый контекст.

print STDOUT $s; #Печатаем в STDOUT (стандартный поток вывода)Более информативный пример, в котором typeglob используется для передачи файлового дескриптора в функцию:

my $file = 'file.txt'; #имя файла, который читаем

open FH, $file; #открываем файл

my $text = \_parseFile(\*FH); #передаем файловый дескриптор в функцию

print $text;

close FH; #обязательно закрываем файловый дескриптор

sub \_parseFile

{

my $fh = shift; #получаем файловый дескриптор

my $text = join('',<$fh>);#построчно читаем из файлового дескриптора, в строку и помещаем в переменную $text

return $text;

}В последних версиях Perl появилась возможность хранить файловые дескрипторы в скалярах, и в новых программах предпочтительно пользоваться именно этим способом.

Также таблицы символов используются для связи двух имён переменной с одним значением в памяти, создавая синонимы, которые могут быть использованы для доступа и модификации значения точно также как и оригинальное имя. Эта возможность является основой системы загружаемых модулей, [классов и объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в Perl.

### Классы и объекты**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=17)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=17)**]**

Объект в Perl представляет собой просто ссылку, связанную с определённым классом (пакетом). Для связи используется функция bless. Подпрограммы такого связанного пакета являются методами и получают ссылку в качестве первого аргумента.

Специальный синтаксис для конструкторов отсутствует, используется подпрограмма пакета, возвращающая ссылку, связанную с этим пакетом.

Также существуют множество расширений для системы ООП, позволяющих менять синтаксис и использовать синтаксический сахар, в числе которых стоит назвать cpan-модуль [Moose](https://ru.wikipedia.org/wiki/Moose_(Perl)" \o "Moose (Perl)) и его облегчённую версию [Moo](https://ru.wikipedia.org/w/index.php?title=Moo_(Perl)&action=edit&redlink=1" \o "Moo (Perl) (страница отсутствует))[[36]](https://ru.wikipedia.org/wiki/Perl#cite_note-36). Moose используется многими проектами и крупными компаниями, такими как [BBC](https://ru.wikipedia.org/wiki/BBC) или [Cisco](https://ru.wikipedia.org/wiki/Cisco" \o "Cisco)[[37]](https://ru.wikipedia.org/wiki/Perl#cite_note-37).

### Регулярные выражения**[**[**править**](https://ru.wikipedia.org/w/index.php?title=Perl&veaction=edit&section=18)**|**[**править код**](https://ru.wikipedia.org/w/index.php?title=Perl&action=edit&section=18)**]**

Важной частью Perl являются [регулярные выражения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F). Благодаря этому Perl хорошо подходит для обработки текстов. Бо́льшая часть работы с регулярными выражениями производится при помощи операторов =~, m// и s///.

Оператор m// используется для проверки на совпадение. В простейшем случае результат выражения $x =~ m/abc/ будет истинным тогда и только тогда, когда строка $x будет соответствовать регулярному выражению abc. Например:

|  |  |
| --- | --- |
| **Пример** | **Значение** |
| $x =~ /abc/ | Строка $x содержит (под)строку «abc». Начальная буква «m» оператора при использовании // может быть опущена. |
| $x =~ m/a(.{1,3})c/ | Строка $x содержит букву «a», затем от одного до трёх любых символов кроме символа перевода строки «\n», и затем букву «c». |
| $x =~ m{^p(erl|ugs)$}i | Строка $x строго равна «perl» или «pugs» без учёта регистра. Так же, вместо // регулярное выражение окружено в {}. |

Поиск и замена выполняются при помощи оператора s///. Конструкция $x =~ s/abc/def/; заменит первое вхождение регулярного выражения abc на строку def.

|  |  |
| --- | --- |
| **Пример** | **Значение** |
| $x =~ s/abc/def/g; | Все вхождения (на что указывает флаг /g — global) подстроки «abc» в $x будут заменены на «def». |
| $x =~ s/a(.{1,3})c/!$1!/; | Первое вхождение в $x буквы «a», затем от одного до трех любых символов кроме символа перевода строки «\n», и затем буквы «c» будет заменено на эти символы между «a» и «c», окружённые «!». Например, «syntactic» станет «synt!cti!». |
| $x =~ s{^p(erl|ugs)}{"P" . lc $1}ieg; | Здесь показан пример использования модификатора /e, указывающего на то, что вместо строки замены будет написан код, результат выполнения которого надо использовать. Все вхождения «perl» или «pugs» в любом регистре будут заменены на «Perl» и «Pugs», соответственно. |

С появлением динамических регулярных выражений (??{ код Perl }) и конструкции для включения в регулярное выражение исполняемого кода Perl (?{ код Perl }) возможности по поиску и замене стали практически безграничными. К примеру, стал возможен поиск конструкций произвольного уровня вложенности.

Регулярные выражения Perl настолько популярны, что они включены напрямую в другие языки, такие как [PHP](https://ru.wikipedia.org/wiki/PHP) и [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript), а также существуют подключаемые библиотеки, реализующие использование выражений в компилируемых языках.