Глава 2. Структура программы

Учебник по Perl

Содержание [скрыть]

Изучение любого языка программирования начинается с его синтаксиса, одну из неотъемлемых частей которого составляет описание структуры программы, определяющей состав и порядок расположения разнообразных конструкций в теле программы. Мы не будем отступать от сложившихся традиций и объясним необходимые понятия на примере простой программы Perl, получающей информацию от пользователя и в ответ печатающей на экране монитора приветствие.

Простая программа

Язык Perl – достаточно простой язык программирования, семантика ключевых слов которого соответствует их значению в английском языке, поэтому даже начинающий его изучение программист, во всяком случае, так утверждают его разработчики, без особого труда может разобраться в простой программе Perl. Ну, что ж, может быть так оно и есть, но, как говорится, "лучше один раз увидеть, чем сто раз услышать". Итак, в примере 2.1 приведен текст программы, которая печатает на экране монитора приглашение ввести имя пользователя, а в ответ просто приветствует его.

Пример 2.1. Простая программа-приветствие

```
01 #! /bin/usr/Perl

02

03 print "Ваше имя?\n";                # Приглашение ввести имя.

04 $name = <STDIN>;                    # Ввод имени с клавиатуры.

05
```

```
06 $~ = NAME_FORMAT;                # Назначение формата вывода.

07 write;                           # Вывод приветствия.

08

09 $~ = NAME_FORMAT_BOTTOM;  # Вывод нижней разделительной черты.

10 write;

11

12 format NAME_FORMAT=              # Начало описания формата.

13 Привет, @»»»»»»»!                # Строка вывода.

14 $name                            # Переменная, значение которой

                                    # подставляется в строку вывода.

15 .                                # Завершение описания формата.

16

17  format NAME_FORMAT_TOP=    # Заголовок формата NAME_FORMAT.

18  ================================

19  Сообщение Pel-программы

20

21 .

22

23  format NAME_FORMAT_TOP=  # Формат вывода нижней разделительной черты
```

```
24 ==============================

25 .
```

Эта программа не совсем типичная для Perl-программы – в ней отсутствуют операторы ветвления, цикла, определение собственных подпрограмм и их вызов и многое другое, но она все же отражает предназначение языка Perl как языка генерирования отчетов, осуществляя вывод приветствия в соответствии с технологией создания отчетов путем определения формата вывода, который в большинстве современных языков либо отсутствует вообще, либо практически не используется программистами.

Если читатель по тексту программы уже понял, как она будет работать (надеемся, что наши комментарии помогли ему в этом), то теперь самое время проверить его догадку – выполнить эту программу. Но прежде следует в обычном текстовом редакторе набрать ее текст и сохранить в файле с расширением pl. Программы Perl являются обычными текстовыми файлами и для их создания можно использовать любой текстовый редактор: в UNIX, например, всегда доступные vi и ed, а в Windows – Блокнот (notepad.exe) или редактор программы Far.

> **Замечание**
>
> Обычно расширение файла (до трех символов) используется для идентификации файлов определенной группы: выполнимые файлы программ (EXE), файлы текстового процессора Word (DOC) и т. д. Для сценариев Perl принято использовать двухбуквенное расширение pl.

Для выполнения программы примера 2.1, сохраненной в файле programl.pl, в любой операционной системе, имеющей командную строку, достаточно набрать в ней команду:

```
Perl program1.pl
```

В результате выполнения программы на экране монитора появится приглашение ввести имя и после ввода имени отобразится приветствие Perl-программы. Дамп экрана после выполнения нашей программы можно увидеть в примере 2.2, где мы полужирным шрифтом выделили ввод пользователя.

Пример 2.2. Вывод программы примера 2.1

```
Ваше имя?
Александр
========================
Сообщение Perl-программы
```

Привет, Александр!

==========

Соответствует ли он вашим соображениям относительно работы этой программы при анализе ее текста? Если да, то мы вас поздравляем, если нет – не надо отчаиваться, несколько наших комментариев поставят все на свои места.

Первый оператор – это специальный комментарий, который для системы UNIX определяет местонахождение интерпретатора Perl. Дело в том, что в этой операционной системе можно сделать любой файл выполняемым с помощью команды

```
chmod +x programl.pl
```

и запустить на выполнение из командной строки:

```
./programl.pl
```

В этом случае первая строка программы сопоставляет файлу приложение, которое должно быть загружено для его обработки. Более того, в ней можно определить при необходимости параметры, или ключи, определяющие режим работы приложения, в нашем случае интерпретатора Perl. Можно задать отображение предупреждающих сообщений или загрузку отладчика в случае обнаружения серьезной ошибки с помощью следующей строки:

```
#! /bin/usr/Perl -w -d
```

(О режимах работы интерпретатора Perl и соответствующих ключах см. главу 14.)

> **Замечание**
>
> При работе в операционной системе Windows можно ассоциировать с расширением файла определенную программу, которая будет вызываться при двойном щелчке мышью на любом файле с таким расширением. Программа установки интерпретатора языка Perl фирмы ActiveState Tool Corp., который называется Active Perl, автоматически устанавливает соответствие файлов с расширением pl и интерпретатора языка Perl. При двойном щелчке на файле сценарий действительно выполняется в окне DOS, которое, однако, автоматически закрывается после выполнения последнего оператора, что не позволяет просмотреть отображенные в нем результаты. Чтобы этого не происходило, следует заменить строку вызова приложения, генерируемую программой установки, на следующую:

```
C:\WINDOWS\Dosprmpt.pif /cПолный__путь\Perl.exe
```

Для этого следует выполнить команду Параметры меню Вид программы Проводник, в диалоговом окне Параметры на вкладке Типы файлов в списке Зарегистрированные типы выделить Perl File и нажатием

кнопки Изменить отобразить диалоговое окно Изменение свойств типа файлов. В этом окне в списке Действия выбрать Open и нажать кнопку Изменить. В появившемся диалоговом окне Изменение действия для типа: Perl File в поле Приложение, исполняющее действие: ввести указанную строку, задав в ключе /c полный путь к папке, где расположен двоичный исполняемый файл интерпретатора Active Perl, который называется Perl.exe.

При работе с интерпретатором Active Perl нет необходимости задавать первую строку, однако если необходимо установить режимы работы интерпретатора, то ее следует задать, причем не обязательно указывать полный путь расположения интерпретатора, достаточно только его имя Perl и необходимые ключи.

> **Внимание**
>
> Все примеры программ в этой книге проверены в интерпретаторе Active Perl 520 для Windows 95/NT и встроенном интерпретаторе Perl системы Linux RedHat 6.1, соответствующих текущей стабильной версии Perl 5.005.

Вообще любой комментарий в языке Perl начинается с символа "#" и распространяется до конца строки. Единственное исключение – это если сразу же после символа комментария следует символ "!".

> **Внимание**
>
> В строке специального комментария не следует вводить ничего, кроме пути местонахождения интерпретатора и его ключей. В противном случае можно получить сообщение об ошибке.

В строке 3 функция print посылает на системное устройство вывода (обычно это монитор компьютера) содержимое списка своих параметров, при необходимости преобразуя его в символьное представление. В нашей программе при выполнении этого оператора на экране монитора отобразится содержимое строки, передаваемой функции print в качестве параметра. Для тех, кто не знаком с языком C, последовательность символов "\n" может показаться странной, тем более, что в дампе экрана (см. пример 2.2) она даже не отображается. Это одна из так называемых управляющих или ESC-последовательностей, которые предназначены для представления неотображаемых символов – в. данном случае символа перехода на новую строку.

Оператор строки 4 ожидает ввод со стандартного устройства ввода (обычно это клавиатура) и присваивает переменной $name введенное пользователем имя. Забегая вперед, скажем, что при работе с файлами в Perl определяются дескрипторы файлов, представляющие собой обычные идентификаторы, используемые в программе для ссылки на файлы, с которыми они ассоциированы. В языке имеется несколько предопределенных дескрипторов файлов, одним из которых является STDIN, ассоциированный со стандартным устройством ввода. Операция о выполняет ввод из файла, заданного своим дескриптором.

Оператор строки 6 назначает системной переменной $~ имя используемого формата для выполнения вывода на стандартное устройство вывода системной функцией write из строки 7. Сам формат задается в строках 12-15 оператором format, в котором определяется имя формата (NAME_FORMAT), завершающееся символом равенства "=". В последующих строках до строки 15, содержащей единственный символ точка ".", фиксирующий завершение объявления формата, задается сам формат, который обычно представляет собой повторяющуюся последовательность двух строк: строки вывода и строки, перечисляющей через запятую переменные, чьи значения при выводе подставляются вместо поледержателей, объявленных в строке вывода. Строка вывода представляет собой именно строку, которая выводится функцией write в определяемый ее параметром файл (если параметр не задан, то вывод осуществляется на стандартное устройство вывода). В этой строке значащими являются все пробельные символы (сам пробел, символы перехода на новую строку и страницу, символ табуляции). Поледержатель – это специальная конструкция в строке вывода, начинающаяся с символа "@", за которым следует определенное количество специальных символов, задающих длину поля, в которое выводится значение переменной, ассоциированной с поледержателем и определенной в списке переменных, задаваемом в строке, непосредственно следующей за строкой вывода. Если в строке вывода поледержателей нет, то строку со списком ассоциированных с ними переменных задавать не надо.

В формате NAME_FORMAT определена одна строка вывода с одним поледержателем, который резервирует поле длиной в 12 символов и определяет, что выводимое значение должно быть прижато вправо (символ ">"). Это означает, что если значение ассоциированной с этим поледержателем переменной $name будет меньше 12 символов, то в этом поле при выводе они будут выровнены по правому краю. Если выводимое значение преобразуется в строку длиной более 12 символов, она обрезается по правому краю, т. е. отображаются первые 12 символов, считая слева направо.

Если для формата определен формат с таким же именем и суффиксом _TOP, то этот формат определяет вид заголовка страницы, который будет отображаться всякий раз при выводе новой страницы с использованием основного формата. Для формата NAME_FORMAT определен формат заголовка в строках 17-19.

Завершается вывод приветствия печатанием разделительной черты функцией write строки 10 с использованием формата NAME_FORMAT_BOTTOM. Обратите внимание, что для использования нового формата вывода нам пришлось изменить значение системной переменной $~. Это следует делать всякий раз, когда необходимо организовать вывод с помощью нового формата.

Итак, мы разработали и выполнили нашу первую программу на языке Perl. Теперь пришло время обратиться к синтаксису языка и рассказать в самых общих чертах, из чего состоит программа на языке Perl, т. е. описать структуру программы.

Объявления и комментарии

Программа Perl представляет собой последовательность операторов и объявлений, которые обрабатываются интерпретатором в том порядке, как они появляются в тексте программы. Во многих языках программирования обязательны объявления всех используемых переменных, типов и других объектов. Синтаксис Perl является "демократичным" в этом отношении и предписывает обязательные объявления только форматов и подпрограмм.

Объявления форматов и подпрограмм являются глобальными, а это означает, что они "видимы" из любого места сценария. Объявления могут располагаться в программе в любом месте, куда можно поместить оператор, но обычно их размещают либо в начале, либо в конце программы, чтобы быстро найти и откорректировать в случае необходимости. Объявления обрабатываются во время компиляции и не влияют на выполнение последовательности операторов, когда откомпилированный во внутренний код интерпретатора сценарий начинает свою работу.

В Perl нет специального оператора объявления переменной, она определяется при первом ее использовании, причем с помощью ключа -w интерпретатора можно задать режим отображения предупреждающих сообщений при попытке использования не инициализированной переменной. Переменные можно определять как глобальные, видимые из любой точки программы, так и с помощью функции ту как локальные, видимые в определенной части программы – блоке.

(Объявления локальных переменных описываются в главе 3 и в главе 11)

Можно объявить подпрограмму с помощью оператора sub, не определяя ее, т. е. не задавая операторы, реализующие ее функцию. После такого объявления подпрограммы ее имя можно использовать как операцию, действующую на список, определяемый передаваемыми ей параметрами.

(Более подробно объявления и определения подпрограмм рассматриваются в главе 11)

Как уже отмечалось выше, если интерпретатор встречает символ #, то он игнорирует любой текст, расположенный за ним. Такая конструкция называется комментарием и ее действие распространяется до конца строки после символа #. Комментарии используются для документирования программы и не следует ими пренебрегать, так как они вносят ясность в понимание того, что выполняет программа. Однако и злоупотреблять ими не следует, так как слишком большое их количество может ухудшить читаемость программы – одну из важных характеристик любой программы.

Комментарии располагаются в любом месте программы. Их можно разместить непосредственно после оператора в той же самой строке, как в нашем примере 2.1, или занять ими всю строку, если первым символом в ней является символ комментария #. Если необходимо временно исключить из потока выполняемых операторов какой-либо из них, то его можно просто закомментировать, не забыв, правда, удалить символ комментария, когда этот оператор снова надо будет включить в поток вычислений.

Выражения и операторы

Оператор – это часть текста программы, которую интерпретатор преобразует в законченную инструкцию, выполняемую компьютером. С точки зрения синтаксиса языка (способов составления правильных конструкций, распознаваемых интерпретатором) оператор состоит из *лексем* – минимальных единиц языка, которые имеют определенный смысл для интерпретатора. Под минимальной единицей языка понимается такая его единица, которая не может быть представлена более мелкими единицами при дальнейшем ее синтаксическом разборе. В языке Perl лексемами могут быть идентификаторы, литералы, знаки операций и разделитель.

Мы дадим определения всем допустимым в языке лексемам. Хотя их семантика (смысл) может оказаться для начинающих программистов и не совсем ясна, но мы вернемся к некоторым определениям в последующих главах, где уточним их и синтаксис, и семантику в связи с вводимыми элементами языка. Дело в том, что, к сожалению, невозможно описать язык без ссылок вперед.

Идентификатор – это последовательность букв, цифр и символа подчеркивания "_", начинающаяся с буквы или подчеркивания и используемая для именования переменных, функций, подпрограмм, дескрипторов файлов, форматов и меток в программе. Программист может использовать любые правильные идентификаторы для именования перечисленных объектов программы, если только они не совпадают с *ключевыми словами* языка — предопределенными идентификаторами, которые имеют специальное значение для интерпретатора языка Perl, например if, unless, goto и т. д. Примеры правильных и неправильных идентификаторов представлены в примере 2.3.

Пример 2.3. Правильные и неправильные идентификаторы

```
# Правильные идентификаторы
myName1
my_Name1
_myName__1

 # Неправильные идентификаторы
1myName   # Начинается с цифры.
-myName   # Начинается не с символа буквы или подчеркивания.
 my%Name   # Используется недопустимый для идентификаторов символ
my       # my является зарезервированным словом.
```

Замечание

Забегая вперед, скажем, что так как имена переменных Perl начинаются со специального символа ("$", "@", "%"), определяющего их тип, после которого следует идентификатор, то в этом случае использование идентификатора, совпадающего с ключевым словом Perl, является правомочным и не вызывает ошибку интерпретатора. Так, следующие имена переменных являются допустимыми: $print, @do, %if, однако подобная практика не рекомендуется. Это замечание не

относится к идентификаторам, используемым для именования дескрипторов файлов и меток, имена которых не начинаются с определенных символов.

(Как используются идентификаторы для объявления переменных см. главу 3.)

(Как используются идентификаторы в дескрипторах файлов см. главу 7.)

(Как используются идентификаторы для объявления форматов см. главу 8.)

Литерал, или *буквальная константа,* – символ или слово в языке программирования, определяющие в отличие от переменной свое собственное значение, а не имя другого элемента языка. Буквальные константы тесно связаны с типами данных, представимыми в языке, и являются, собственно говоря, их представителями. В Perl литералами являются числа и строки.

```
123 # Целое число.
23.56 # Вещественное число с фиксированной точкой.
2E+6 # Вещественное число с плавающей точкой.
"Язык Perl" # Строковый литерал.
```

(О литералах см. в главе 3.)

Знаки операций – это один или более специальных символов, определяющих действия, которые должны быть выполнены над величинами, называемыми операндами. Выполняемые действия называются операциями, которые могут быть унарными (применяются к одному операнду), бинарными (применяются к двум операндам) и тернарные (участвуют три операнда).

Пример 2.5. Операции языка Perl

```
++$n; # Унарная операция (++)
23 * $n; # Бинарная операция (*)
$n >= 3 ? print "true" : print "false"; # Тернарная операция (?:)
```

(Об операциях и используемых знаках операций см. в главе 4.)

Разделитель – это символ ";", которым завершается любой оператор и который сообщает об этом интерпретатору. Использование разделителя позволяет на одной строке задавать несколько операторов, хотя это и не принято в практике программирования, так как ухудшает читаемость текста программы.

В операторе все его лексемы могут отделяться любым числом *пробельных символов,* к которым относятся сам пробел, знак табуляции, символ новой строки, возврат каретки и символ перехода на новую строку. Поэтому один оператор можно записать на нескольких строках и для этого не надо использовать никакого символа продолжения, требующегося в других языках программирования. Например, оператор номер 4 присвоения данных, введенных с клавиатуры, из примера 2.1 можно записать и так:

Пример 2.6. Использование пробельных символов в операторе

```
$name
= <STDIN>
;
```

Можно вообще не использовать пробельные символы в операторе, но для обеспечения читаемости программы мы рекомендуем отделять лексемы одним пробелом, тем более что могут встречаться ситуации, когда интерпретатор не однозначно выделяет лексемы из непрерывного потока символов.

Замечание

Так как пробельные символы не являются значащими в Perl, то обычно их используют для структуризации текста программы, которая заключается в написании некоторой группы операторов, логически подчиненных некоторой конструкции языка, с некоторым отступом относительно этой конструкции. Например, подобный подход можно применить к блоку операторов, выполняющихся в конструкции цикла, сдвинув все их вправо относительно этой конструкции. Структуризация текста программы способствует ее лучшему прочтению и широко практикуется программистами многих языков программирования, например языка C.

Операторы в языке Perl могут быть простыми или составными. *Простой оператор* — это выражение, завершающееся разделителем точкой с запятой ",-", и которое вычисляется исключительно ради своего побочного эффекта. Что такое побочный эффект выражения мы определим немного позже, а сейчас остановимся на понятии "выражение".

Выражение — последовательность литералов, переменных и функций, соединенных одной или более операцией, которые вычисляют скаляр или массив, т. е. при обработке интерпретатором выражения единственным действием является *вычисление* значения, а не выполнение некоторых других действий, например присвоение переменной нового значения. При вычислении выражения могут проявляться *побочные эффекты,* когда при вычислении выражения меняется значение переменной, входящей в выражение. Они могут вызываться, например, операциями увеличения (++) и уменьшения (--) или при вызове функции, которая изменяет значение своего фактического параметра. Как упоминалось выше, простой оператор и выполняется, чтобы реализовать этот побочный эффект, иначе какой смысл просто вычислить выражение, значение которого никоим образом нельзя использовать.

Пример 2.7. Простые операторы

```
++$n; # Значение переменной $n увеличивается на единицу.
123*$n; # Простой оператор без побочного эффекта.
```

Каждый простой оператор может иметь модификатор, который располагается после выражения перед завершающей точкой с запятой. Модификаторы представляют собой ключевые слова if, unless, while и until, за которыми следует некоторое выражение. Семантика использования модификатора

заключается в том, что простой оператор выполняется, если истинно или ложно выражение, стоящее после модификатора, в зависимости от используемого модификатора. Модификаторы употребляются так же, как и в обычном разговорном английском языке. Например, простой оператор

```
$n++ while <STDIN>;
```

будет выполняться, увеличивая всякий раз значение переменной $n на единицу, пока пользователь будет осуществлять ввод с клавиатуры. Остановить выполнение этого оператора можно вводом комбинации клавиш <Ctrl>+<Z>ium <Ctrl>+<C>.

Замечание

Язык Perl вобрал в себя лучшие элементы других языков программирования, в основном C. Конструкция модификаторов заимствована из умершего языка BASIC/PLUS фирмы Digital Equipment Corp.

(Подробно все модификаторы простых операторов рассматриваются в главе 5.)

Чтобы определить конструкцию, называемую составным оператором, нам придется сначала ввести понятие "блок". Последовательность операторов Perl, определяющая область видимости переменных, называется *блоком.* После знакомства с переменными это определение не будет таким туманным, каким оно может показаться сейчас начинающему программисту. Для целей этой главы достаточно мыслить блок как последовательность операторов, заключенную в фигурные скобки:

```
{ оператор_1;
оператор_n; }
```

Составной оператор определяется в терминах блока и может быть одного из следующих видов:

Пример 2.8. Составные операторы

if (выражение) *БЛОК*

if (выражение) *БЛОК_1* else *БЛОК__2*

if (выражение_1) *БЛОК_1* elsif (выражение_2) *БЛОК_2 ...* else *БЛОК_п*

METKA while (вьражение) *БЛОК*

METKA while (вьражение) *БЛОК_1* continue *ЕЛОК_2*

METKA for (вьражение_1; выражение_2; аьражение_3) *БЛОК*

METKA foreach *переменная (список) БЛОК .METKA БЛОК_1* continue *БЛОК_2*

Обратим внимание читателя на то, что, в отличие от языков программирования C и Pascal, составные операторы Perl определяются в терминах блоков, а не в терминах операторов. Это означает, что там, где нужен блок, он всегда должен задаваться с помощью фигурных скобок. В составных операторах, если даже блок состоит из одного оператора, он должен быть заключен в фигурные скобки. Такой синтаксис не приводит к двусмысленностям и, например, во вложенных

операторах условия всегда ясно, с каким if согласуется else или elsif. Метка, представляющая собой идентификатор с двоеточием ":", в составных операторах не обязательна, но если она присутствует, то имеет значение для операторов управления циклами next, last и redo.

(Подробно все составные операторы рассматриваются в главе 5.)

* * *

В этой главе мы познакомились с основными синтаксическими понятиями, используемыми для формирования правильных конструкций языка Perl. Узнали из каких основных элементов состоит Perl-программа, а также разработали и выполнили нашу первую программу.

Вопросы для самоконтроля

1. Что такое лексемы и какими элементами они представлены в языке Perl?
2. Что такое выражение и зачем оно создается?
3. Чем выражение отличается от оператора?
4. Каковы различия между объявлением и оператором?
5. Какие классы операторов имеются в языке Perl?
6. Что представляет собой Perl-программа?

Упражнения

1. Модифицируйте программу примера 2.1, чтобы она всегда приветствовала весь мир, т. е. при ее выполнении всегда отображалось бы "Привет, мир!".
2. Модифицируйте программу примера 2.1, использовав для вывода оператор print, имеющий следующий синтаксис:
   print выражение!, выражение2, . . . , выражениеn;
3. Исправьте ошибку в программе:
   #! /usr/bin/Perl Это строка показывает местонахождение интерпретатора print "Perl";
4. Исправьте ошибки в программе:
   i! /usr/bin/Perl -w
   $write = 24
   $two =3
   $rez = $write * $two
5. Исправьте ошибку в программе:
   '#! /usr/bin/Perl -w $write = 24;
   # Печать переменной! print $write;