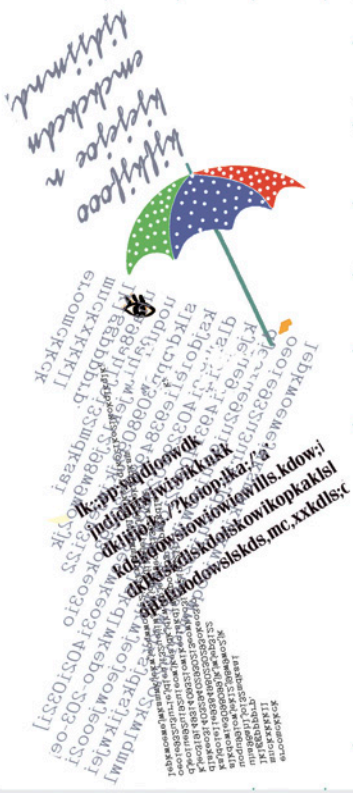
bhv
www.bhv.ru
www.bhv.kiev.ua

А. В. Матросов
М. П. Чаунин

Perl

Самоучитель

- Пакеты, модули, библиотеки
- Объектно-ориентированное программирование
- CGI-сценарии



Современные средства программирования и системного администрирования

Александр Матросов,
Михаил Чаунин

САМОУЧИТЕЛЬ Perl

Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

В книге изложены основы современного языка Perl, популярность которого постоянно возрастает, особенно в таких областях, как обработка текста, CGI-программирование, системное администрирование. Язык описан по схеме от простого к сложному: типы данных, переменные, операции, операторы и т. д. Рассматривается объектно-ориентированная технология программирования. Приведенные в книге примеры и упражнения, которые авторы реализовали на различных платформах, помогут читателю разобраться в изложенном материале.

Для широкого круга пользователей

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Наталья Таркова*
Редакторы	*Татьяна Кручинина, Ольга Афанасьева*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Светлана Журавина*
Дизайн обложки	*Ангелины Лужиной*
Зав. производством	*Николай Тверских*

Матросов А. В., Чаунин М. П.

Самоучитель Perl. — СПб.: БХВ — Санкт-Петербург, 2000. — 432 с.: ил.

ISBN 5-8206-0070-3

Лицензия ЛР № 065953 от 15.06.98. Подписано в печать 20.04.2000.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 34,83.
Тираж 5000 экз. Заказ
"БХВ — Санкт-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9 линия, 12.

Содержание

Предисловие **1**

Глава 1. Введение в мир Perl **5**
1.1. История языка Perl 6
1.2. Характерные черты Perl 8
1.3. Области применения Perl 12
Системная поддержка UNIX 13
CGI-сценарии 13
Обработка почты 14
Поддержка узлов Web 14
Вопросы для самоконтроля 15

Глава 2. Структура программы **16**
2.1. Простая программа 16
2.2. Объявления и комментарии 21
2.3. Выражения и операторы 22
Вопросы для самоконтроля 27
Упражнения 27

Глава 3. Типы данных **29**
3.1. Алфавит языка 29
3.2. Скалярный тип данных 31
3.3. Массивы скаляров 42
3.4. Ассоциативные массивы 47
3.5. Переменные 52
Вопросы для самоконтроля 55
Упражнения 56

Глава 4. Операции и выражения **57**
4.1. Арифметические операции 58
4.1.1. Бинарные арифметические операции 58
4.1.2. Унарные арифметические операции 60
4.1.3. Операции увеличения и уменьшения 61
4.2. Операции конкатенации и повторения 62
4.3. Операции отношения 65
4.3.1. Числовые операции отношения 65
4.3.2. Строковые операции отношения 67
4.4. Логические операции 68
4.5. Побитовые операции 70
4.5.1. Числовые операнды 70
4.5.2. Строковые операнды 73
4.6. Операции присваивания 74
4.7. Ссылки и операция разыменования 77
4.8. Операции связывания 78
4.9. Именованные унарные операции 79
4.10. Операции ввода/вывода 79
4.10.1. Операция print 80
4.10.2. Выполнение системных команд 80
4.10.3. Операция <> 80
4.11. Разные операции 81
4.11.1. Операция диапазон 81
4.11.2. Операция запятая 86
4.11.3. Операция выбора 87

4.12. Списковые операции ... 88
4.13. Операции заключения в кавычки ... 88
4.13.1. Операция q{ } ... 89
4.13.2. Операция qq{ } ... 90
4.13.3. Операция qx{ } ... 90
4.13.4. Операция qw{ } ... 91
4.13.5. Операция "документ здесь" ... 91
4.14. Выражения ... 94
4.14.1. Термы ... 95
4.14.2. Приоритет операций ... 95
4.14.3. Контекст ... 100
Вопросы для самоконтроля ... 102
Упражнения ... 102

Глава 5. Операторы ... 104
5.1. Простые операторы ... 104
5.2. Модификаторы простых операторов ... 105
5.2.1. Модификаторы if и unless ... 106
5.2.2. Модификаторы while и until ... 107
5.2.3. Модификатор foreach ... 109
5.3. Составные операторы ... 110
5.3.1. Блоки ... 111
5.3.2. Операторы ветвления ... 113
5.4. Операторы цикла ... 116
5.4.1. Циклы while и until ... 116
5.4.2. Цикл for ... 118
5.4.3. Цикл foreach ... 122
5.5. Команды управления циклом ... 125
5.5.1. Команда last ... 126
5.5.2. Команда next ... 128
5.5.3. Команда redo ... 129
5.6. Именованные блоки ... 131
5.7. Оператор безусловного перехода ... 133
Вопросы для самоконтроля ... 134
Упражнения ... 135

Глава 6. Операции ввода/вывода ... 137
6.1. Операция ввода команды ... 137
6.2. Операция <> ... 140
6.3. Функция print ... 145
Вопросы для самоконтроля ... 147
Упражнения ... 147

Глава 7. Работа с файлами ... 148
7.1. Дескрипторы файлов ... 148
7.2. Доступ к файлам ... 151
7.3. Операции с файлами ... 162
7.4. Получение информации о файле ... 168
7.5. Операции с каталогами ... 171
Вопросы для самоконтроля ... 174
Упражнения ... 174

Глава 8. Форматы ... 175
8.1. Объявление формата ... 175
8.2. Использование нескольких форматов ... 182
Вопросы для самоконтроля ... 186

Глава 9. Ссылки 187
9.1. Виды ссылок 187
9.2. Создание ссылок 190
9.2.1. Операция ссылки "" 190
9.2.2. Конструктор анонимного массива 191
9.2.3. Конструктор анонимного ассоциативного массива 191
9.2.4. Другие способы 192
9.3. Разыменование ссылок 194
9.3.1. Разыменование простой скалярной переменной 195
9.3.2. Блоки в операциях разыменования ссылок 195
9.3.3. Операция разыменования "->" 196
9.4. Символические ссылки 198
9.5. Использование ссылок 200
9.5.1. Замыкания 201
9.5.2. Массив массивов 203
9.5.3. Другие структуры данных 204
Вопросы для самоконтроля 210
Упражнения 210
Глава 10. Работа со строками 212
10.1. Регулярные выражения 212
10.1.1. Метасимволы 213
10.1.2. Метапоследовательности 215
10.1.3. Атомы 217
10.1.4. Обратные ссылки 218
10.1.5. Расширенный синтаксис регулярных выражений 220
10.1.6. Сводка результатов 223
10.2. Операции с регулярными выражениями 226
10.2.1. Операция поиска 226
10.2.2. Операция замены 231
10.2.3. Операция транслитерации 231
10.2.4. Операция заключения в кавычки qr// 233
10.3. Функции для работы со строками 234
Вопросы для самоконтроля 245
Упражнения 247
Глава 11. Подпрограммы и функции 249
11.1. Определение подпрограммы 249
11.2. Вызов подпрограммы 251
11.3. Локальные переменные в подпрограммах 252
11.3.1. Функция my() 253
11.3.2. Функция local() 253
11.4. Передача параметров 255
11.4.1. Передача по ссылке параметров-массивов 256
11.5. В каких случаях функцию local нельзя заменить функцией my 260
11.6. Прототипы 263
11.7. Рекурсивные подпрограммы 265
Вопросы для самоконтроля 266
Упражнения 267
Глава 12. Пакеты, библиотеки, модули 268
12.1. Пакеты 268
12.1.1. Таблицы символов 271
12.1.2. Конструктор и деструктор пакета BEGIN и END 272
12.1.3. Автозагрузка 274

12.2. Библиотеки ... 275
12.2.1. Функция require() ... 275
12.2.2. Создание и подключение библиотечного файла ... 277
12.3. Модули ... 278
12.3.1. Функция use() ... 279
12.3.2. Создание и подключение модуля ... 280
12.3.3. Функция no ( ) ... 282
12.3.4. Стандартные модули Perl ... 282
12.3.5. Прагма-библиотеки ... 283
Вопросы для самоконтроля ... 285
Упражнения ... 286
Глава 13. Объектно-ориентированное программирование в языке Perl ... 287
13.1. Классы и объекты ... 288
13.2. Методы ... 290
13.2.1. Конструкторы ... 290
13.2.2. Методы класса и методы объекта ... 292
13.2.3. Вызов метода ... 294
13.2.4. Деструкторы ... 297
13.3. Обобщающий пример ... 297
Вопросы для самоконтроля ... 301
Упражнение ... 301
Глава 14. Запуск интерпретатора и режим отладки ... 302
14.1. Опции командной строки ... 302
14.2. Отладчик Perl ... 308
14.2.1. Просмотр текста программы ... 309
14.2.2. Выполнение кода ... 310
14.2.3. Просмотр значений переменных ... 311
14.2.4. Точки останова и действия ... 312
Вопросы для самоконтроля ... 318
Глава 15. Язык Perl и CGI-программирование ... 319
15.1. Основные понятия ... 319
15.2. HTML-формы ... 320
15.2.1. Тэг <FORM> ... 321
15.2.2. Тэг <INPUT> ... 322
15.2.3. Тэг <SELECT> ... 325
15.2.4. Тэг <TEXTAREA> ... 326
15.2.5. Пример формы ... 327
15.3. Передача информации CGI-программе ... 330
15.4. CGI-сценарии ... 332
15.4.1. Переменные среды CGI ... 335
15.4.2. Обработка данных формы ... 337
15.4.3. Пример создания собственного CGI-сценария ... 340
15.4.4. Модуль CGI.pm ... 348
Вопросы для самоконтроля ... 360
Упражнения ... 360
Глава 16. Ресурсы Perl ... 363
16.1. Конференции ... 363
16.2. Специализированные Web-узлы Perl ... 364
16.3. Архив CPAN ... 365
Приложение 1. Стандартные функции Perl ... 368
Приложение 2. Модули Perl ... 406
Приложение 3. Специальные переменные ... 417
Предметный указатель ... 425

Предисловие

Предисловие, на наш взгляд, должно дать читателю информацию, на основании которой он решает, нужна ли ему эта книга.

О чем наша книга. Она, естественно, о языке Perl, потому что так заявлено в названии. Кому он нужен, этот Perl? Тем, кто создает CGI-сценарии, занимается администрированием системы при помощи написания скриптов, а не щелкая левой кнопкой мыши, обратывает тексты, решает многие другие задачи из смежных областей и при этом нуждается в мощном, но простом в применении средстве, позволяющем создавать большие программы и маленькие программки и быстро их опробовать. Тем, кто преподает программирование, тоже полезно иметь представление об этом языке, так как он обладает интересными свойствами, отсутствующими в традиционных языках программирования, используемых в процессе обучения.

Нам нравятся некоторые особенности языка: зависимость результата от контекста, ассоциативные массивы, тип данных typeglob, пакеты, реализация объектно-ориентированного программирования и, конечно, средства обработки текста. Если вам не интересно хотя бы узнать, что все это означает, то можете книгу отложить. Если все перечисленное вам уже известно, то тоже можете ее отложить, потому что эта книга для тех читателей, кто еще только начинает изучать Perl самостоятельно.

Язык Perl создан системным программистом Ларри Уоллом (Larry Wall) как средство UNIX, позволяющее "склеивать" из программок, выполняющих отдельные функции, большие сценарии для решения комплекса задач, связанных с администрированием, обработкой текста и т. д. В дальнейшем он вышел за эти рамки, превратился в настоящий язык программирования, в котором нашли отражение многие тенденции, обозначившиеся в технологии программирования за последнее десятилетие, и получил широкое распространение в связи с развитием Inernet. Perl является основным средством создания приложений CGI, удобен для решения задач администрирования Web-серверов, электронной почты и других систем. Благодаря быстроте и легкости написания сценариев на этом языке он распространился и на другие платформы: DOS, Windows, OS/2, Mac, VMS и пр. Одно из основных достоинств языка Perl — его открытость и доступность. В сети Internet мож-

но получить совершенно бесплатно исходные тексты интерпретатора perl (язык Perl — интерпретируемый, что в некоторых случаях является преимуществом) и модулей его расширения.

Данная книга — самоучитель языка Perl, который изучается, что называется, с нуля, т. е. предполагается, что читатель не знаком с этим языком — все необходимое он узнает, последовательно изучив темы и закрепив пройденный материал, отвечая на вопросы и выполняя упражнения, приведенные в конце каждой главы. Повторим, что предлагаемый материал представляет всего лишь основы языка Perl. Наша книга ни в коей мере не претендует на учебник по программированию на языке Perl. В ней вы не найдете методологию программирования или готовые рецепты решения задач, в ней нет подробного описания наиболее часто используемых модулей и решения задач с их помощью, но, прочитав книгу, вы приобретете базовые знания, которые позволят разобраться в любом сценарии Perl.

При описании языка мы придерживались следующей схемы: сначала познакомить читателя с типами данных, затем с допустимыми операциями языка, далее с операторами и потом уже со специальными синтаксическими конструкциями (ссылки, регулярные выражения, подпрограммы, пакеты, модули и объектно-ориентированное программирование), т. е. мы старались идти от простых понятий к более сложным, что, как нам кажется, позволяет читателю получить целостное представление о языке программирования. Завершается изучение языка описанием того, как его использовать для создания CGI-сценариев — области Web-программирования, где он широко распространен.

Что осталось за рамками нашей книги? Очень многое, так как мир Perl велик. Это вопросы взамодействия с базовой операционой системой, межпроцессное взаимодействие в UNIX, средства организации сетей, совместное использование Perl и других языков программирования, особенности работы на платформах, отличных от UNIX, работа с базами данных и еще многое другое. В главе о применении Perl в CGI-программировании неплохо было бы рассказать о протоколе HTTP и конфигурировании Web-сервера. По этим темам можно написать еще одну книгу, как минимум, такого же объема. Мы сознательно исключили данные вопросы из рассмотрения, так как они являются достаточно специфическими.

Все примеры самоучителя рассчитаны на использование версии Perl 5.005, которая в процессе работы над книгой считалась последней устойчивой версией, и проверены в операционных системах Linux, Windows NT 4.0, Windows 95. Перед самым окончанием работы (конец марта 2000 года) появилась новая версия языка Perl 5.6, обладающая дополнительными возможностями по сравнению с предыдущей версией. Однако это не должно расстраивать читателя, поскольку основные понятия остались, естественно, без изменений.

Следует иметь в виду, что "родной" платформой Perl является система UNIX, поэтому результат выполнения некоторых примеров в других операционных системах может не соответствовать тому, что написано в книге.

Используемые обозначения

Курсив применяется для выделения терминов и ссылок на главы и разделы самоучителя.

Все тексты и консольный вывод программ Perl, переменные, операторы, функции и другие элементы языка представлены `моноширинным шрифтом`.

При описании синтаксических конструкций языка применяются следующие условные обозначения:

`[параметр]` — необязательный параметр

`...` — повторение предыдущей конструкции

Литература

Наиболее известной и часто цитируемой книгой по программированию на языке Perl является издание *Larry Wall, Tom Christiansen, Randal Schwartz.— Programming Perl, 2nd Edition. O'Reilly & Associates, 1996, 646 pages.*

Подробная информация содержится также в документации, входящей в дистрибутивный комплект Perl.

Завершая предисловие, мы хотели бы выразить благодарность директору издательства "БХВ — Санкт-Петербург" Вадиму Сергееву и главному редактору Екатерине Кондуковой, которые предоставили нам возможность написать эту книгу.

Авторы

Глава 1

Введение в мир Perl

Что такое Perl? Это сокращенное название языка программирования *P*ractical *E*xtraction and *R*eport *L*anguage (Практический язык извлечений и отчетов). Что подразумевается под "извлечениями" и "отчетами"? Почему *практический* язык? Для чего он предназначен? Какие задачи можно решать с его помощью? Эти и многие другие вопросы возникают, естественно, у любого человека, хоть немного знакомого с информатикой, когда он впервые сталкивается с новым для него языком программирования. Эта глава и задумывалась как ответ на поставленные выше вопросы, так как зная, что может, для чего предназначен язык программирования (а время универсальных языков, кажется, миновало), программист, в конечном счете, решает, а стоит ли тратить время на его изучение. Хотя здесь также встают вопросы о легкости и быстроте освоения нового языка, доступности компиляторов, существовании службы его поддержки, стоимости и т. д. Об этом также пойдет речь в этой главе, которая познакомит читателя с огромным миром Perl-программирования, и станет той отправной точкой, с которой он, мы надеемся, стремительно и без оглядки войдет в него и останется в нем навсегда.

Язык Perl родился в недрах операционной системы Unix как реакция одного талантливого программиста на ограниченную возможность стандартных средств системного администрирования в этой операционной среде. Авторы прекрасно осознают, что большинство читателей знакомы с Unix, возможно, только по названиям книг, лежащих на полках магазинов, так как традиция изучения информационных технологий в нашей стране связана больше с операционными системами семейства Microsoft Windows[1], чем с системой UNIX, которая является базой изучения информатики в западных университетах. Поэтому для воспитанных в традициях Windows читателей мы сделаем небольшое отступление и кратко охарактеризуем процедуру администрирования UNIX, которая радикально отличается от аналогичной работы в операционной системе Windows.

Под *администрированием* понимается настройка операционной системы через установку значений ее параметров таким образом, чтобы она отвечала потребностям отдельного пользователя или группы пользователей. В систе-

[1] Под семейством операционных систем Microsoft Windows понимаются операционные системы Windows 95/98/NT.

мах семейства Windows подобная работа выполняется с помощью Реестра, представляющего собой базу данных двоичных данных, а для изменения параметров используется специальная программа `regedit`. В системе UNIX настройка осуществляется через специальные конфигурационные файлы, являющиеся обычными текстовыми файлами, и все изменения осуществляются выполнением команд, написанных на специальном языке оболочки (shell) и выполняемых, как правило, из командной строки. (Несколько лет назад на персональных компьютерах была широко распространена операционная система MS-DOS фирмы Microsoft, в которой для ввода команд также использовалась командная строка, поэтому читателю, работавшему в этой операционной системе, командная строка знакома.) В системе UNIX пользователь может создавать собственные команды на основе команд интерпретатора shell, сохранять их в обычных текстовых файлах и впоследствии выполнять также, как обычные стандартные команды операционной системы через командную строку. Следует отметить, что оболочка shell операционной системы UNIX является интерпретатором, в связи с чем команды пользователя имеют еще одно название — их называют *сценариями* или *скриптами* (script). Администратору операционной системы UNIX приходится писать большое количество скриптов, которые обрабатывают другие скрипты – текстовые файлы. Для этих целей обычно кроме командного языка оболочки shell используются специальные программы обработки текстовых файлов:

- awk — программа сопоставления с образцами и генератор отчетов;
- sed — пакетный редактор текстовых файлов.

Обе эти программы являются фильтрами, которые последовательно считывают строки входных файлов и выполняют применимые к строке действия, определенные с помощью команд этих программ. Основными действиями являются выделение цепочек символов по заданным шаблонам, замена их по определенным правилам и генерирование новых файлов.

Теперь можно перейти и к объекту нашего изучения — языку Perl, тем более что, как нам кажется, читателю уже должно быть понятно, почему он называется языком извлечений и отчетов. И начнем мы с истории его создания и разработки, которая, по существу, позволяет полнее понять его содержание.

1.1. История языка Perl

Perl был разработан Ларри Уоллом (Larry Wall) в 1986 году, когда он являлся системным администратором одного проекта UNIX, связанного с созданием многоуровневой безопасной сети, объединявшей несколько компьютеров, разнесенных на большие расстояния. Работа была выполнена, но потребовалось создание отчетов на основе большого числа файлов с многочисленными перекрестными ссылками между ними.

Первоначально Ларри предполагал использовать для этих целей фильтр awk, но оказалось, что последний не мог управлять открытием и закрытием большого числа файлов на основе содержащейся в них же самих информации о расположении файлов. Его первой мыслью было написать специальную системную утилиту, решающую поставленную задачу, но вспомнив, что до этого ему уже пришлось написать несколько утилит для решения задач, не "берущихся" стандартными средствами UNIX, он принял кардинальное решение — разработать язык программирования, который сочетал бы в себе возможности обработки текстовых файлов (sed), генерации отчетов (awk), решения системных задач (shell) и низкоуровневое программирование, доступное на языке C. Результатом этого решения и явился язык Perl, интерпретатор для которого был написан на C.

По утверждению самого Ларри Уолла при создании языка Perl им двигала лень — не в прямом смысле, а в смысле того, что для решения стоявшей перед ним задачи следовало бы написать большое количество программ на разных языках, входящих в состав инструментальных средств UNIX, а это достаточно утомительное занятие.

Новый язык программирования сочетал в себе возможности системного администрирования и обработки файлов — две основные задачи, решаемые обычно при программировании в системе UNIX. Причем следует отметить, что язык Perl появился из практических соображений, а не из-за желания создать еще одно "красивое" средство для работы в UNIX, поэтому-то он и получил широкое распространение среди системных администраторов, когда Ларри Уолл предоставил его широкому кругу пользователей. С появлением языка Perl появилась возможность решать задачи с помощью одного инструмента, и не тратить время на изучение нескольких языков среды программирования UNIX.

Первая версия языка не содержала многих возможностей, которые можно найти в последней версии Perl, с которой читатель познакомится в нашей книге и которая идентифицируется как версия 5.005_03 и считается устойчивой. Первоначально язык включал:

- простой поиск по строковым образцам (шаблонам) в файлах;
- дескрипторы файлов;
- скалярные переменные;
- форматы.

Вся документация умещалась на 15 страницах, но Perl решал задачи быстрее, чем sed или awk, и быстро стал использоваться не только для решения задач системного администрирования.

В дальнейшем сам Ларри Уолл позаимствовал у Генри Спенсера (Henry Spencer) пакет для работы с регулярными выражениями и модифицировал его для языка Perl. Другие функциональные возможности были разработаны

не только Ларри Уоллом, но и его друзьями и коллегами, и включены в состав языка. Опубликование в Internet привело к появлению сообщества единомышленников, которые не только эксплуатировали, но и развивали язык. Он и по настоящее время продолжает интенсивно развиваться за счет разработки пакетов, реализующих новые применения языка к развивающимся информационным технологиям. В табл. 1.1 представлена динамика появления новых версий языка Perl, начиная с самой первой:

Таблица 1.1. *Версии языка Perl и даты их выпуска*

Версия	Дата выпуска
perl1	Январь, 1988
perl2	Июнь, 1988
perl3	Октябрь, 1989
perl4	Март, 1991
perl5	Октябрь, 1994

В настоящее время, как уже отмечалось ранее, устойчивой версией считается версия Perl 5.005_03, но уже существует версия 5.005_67. Их все можно получить с основного узла Web, поддерживающего язык Perl, по адресу **http://www.perl.com/**.

Замечание

В литературе по языку Perl принято, ссылаясь на сам язык, писать его с прописной буквой (Perl), а строчными буквами (perl) обозначать интерпретатор языка. По образному высказыванию самого Ларри Уолла: "perl — это ничего более как всего лишь интерпретатор Perl".

1.2. Характерные черты Perl

Perl — это интерпретируемый язык, оптимизированный для просмотра содержимого текстовых файлов, выделения из них информации и генерирования отчетов на основе этой информации, а также просто хороший язык для выполнения многих задач системного администрирования UNIX. Он обладает большим набором преимуществ как язык сценариев общего назначения, которые проявляются через его характерные черты и возможности.

Первым в цепочке достоинств языка Perl мы назовем его *интерпретируемость*. Конечно, некоторые программисты, прочитав это, скажут: "Ну вот, нашли себе достоинство. Посмотрим, как быстро будет выполняться программа Perl длиной, скажем, в тысячу операторов?". Что ж, замечание су-

щественное, если рассматривать Perl как язык создания больших информационных систем, и совершенно не выдерживающее критики, если вспомнить, для чего он предназначен — задач администрирования и обработки текстовых файлов — небольших по размерам сценариев, решающих нетрадиционные задачи, для программирования которых могло бы потребоваться взаимодействие нескольких специализированных языков. Разработка подобных решений с помощью компилируемых языков программирования потребовала бы на много больше времени, чем использование одного интерпретируемого: ведь цикл разработки программ на таком языке короче и проще, чем на компилируемом. Мы постепенно создаем программу, добавляя необходимые операторы, и сразу же получаем результаты, когда она завершена: интерпретатор perl постепенно компилирует все операторы во внутренний байт-код и программа готова к выполнению, как только в ней поставлена последняя точка (точнее точка с запятой, завершающая последний оператор). Для небольших по объему программ — это достаточное преимущество, так как отладка занимает много времени. Да, интерпретируемая программа, естественно, будет выполняться медленнее программы, представленной в формате двоичного файла и выполняющейся без предварительной обработки интерпретатором, но если в этом возникнет необходимость, то можно решение на языке Perl использовать в качестве прототипа для компилируемого языка, например C. Суммируя все сказанное, можно заключить, что Perl позволяет легко и быстро получить требуемое решение задачи, сочетая в себе элементы компилируемых и интерпретируемых языков программирования.

Замечание

Интерпретатор perl, как, вероятно, заметил внимательный читатель, отличается от традиционных интерпретаторов тем, что программа транслируется в промежуточный байт-код, и только после этого выполняется. В традиционных интерпретаторах каждый вводимый оператор интерпретируется и сразу же выполняется, что может приводить к синтаксическим ошибкам во время выполнения. Perl-программа свободна от этого "недостатка", так как все синтаксические ошибки обнаруживаются во время трансляции в байт-код.

Вторым преимуществом использования Perl для решения соответствующих задач (мы имеем в виду сетевые возможности) является его доступность для большинства серверных платформ:

- практически все варианты UNIX;
- MS-DOS;
- Windows NT;
- Windows 95/98;
- OS/2;

❒ Macintosh.

Для всех перечисленных платформ разработаны и свободно распространяются интерпретаторы perl вместе с документацией по их установке и работе, что приятно отличает его от других программных средств. И здесь уместно сказать несколько слов об условиях использования и распространения самого Perl и разработанных на нем программ.

(О том, где можно найти и получить интерпретатор perl, см. главу 16.)

Одним из способов распространения свободно распространяемого программного обеспечения, а именно таков интерпретатор perl, является использование Общей открытой лицензии GNU. По условиям этой лицензии файлы исходного текста программного продукта распространяются совершенно свободно и могут быть использованы любым лицом. Однако любые версии программы, созданные путем модификации этого кода, должны реализоваться также на условиях Общей открытой лицензии GNU, т. е. следует предоставлять файлы исходных текстов нового продукта любому, кто их захочет иметь. Этого зачастую вполне достаточно, чтобы защитить интересы автора первоначального программного продукта, однако может приводить к большому количеству производных версий исходного продукта, что приводит к "отчуждению" автора исходного продукта от процесса модификации его детища. Более того, в связи с большим количеством разнообразных версий, пользователям становится трудно определить, какая версия пакета является на текущий момент окончательной, будут ли написанные им сценарии, если речь идет о perl, правильно работать с имеющейся у него версией, и т. п.

В связи с изложенными недостатками лицензии GNU, интерпретаторы языка Perl выпускаются на условиях лицензии Artistic License (Артистической лицензии), которая является некоторой вариацией лицензии GNU, и ее смысл заключается в том, что любой, кто выпускает пакет, полученный на основе Perl, должен ясно осознавать, что его пакет не является истинным пакетом Perl. Поэтому все изменения должны быть тщательно документированы и отмечены, выполнимые модули, в случае изменения, должны быть переименованы, а исходные модули должны распространяться вместе с модифицированной версией. Эффект от подобных условий заключается в том, что автор первоначального продукта всегда определяется как его владелец. При использовании Artistic License все условия Общей открытой лицензии GNU остаются в силе, т. е. она продолжает применяться.

Третьим преимуществом языка Perl можно назвать его практическую направленность, т. е. он создавался из практических соображений решения задач администрирования и разработки приложений для UNIX, а это означает, что он обладает следующими важными свойствами:

❒ полнотой;

❒ простотой использования;

❒ эффективностью.

Под полнотой Perl понимается его способность решать все возникающие в системе UNIX в связи с ее администрированием задачи. И это действительно так! Ведь язык Perl, как отмечалось выше, вобрал в себя все наилучшие возможности стандартных средств администрирования UNIX, перечисленных в табл. 1.2.

Таблица 1.2. *Стандартные средства администрирования UNIX*

Язык программирования	Характеристика
awk	Язык выделения по образцам информации из текстовых файлов
C	Компилируемый язык общего назначения для решения задач низкого уровня
shell	Основной командный язык запуска программ и скриптов, написанных на других языках программирования
sed	Потоковый редактор обработки текстовых файлов

Эти средства продолжают использоваться, так как каждое из них является прекрасным инструментом для выполнения тех задач, для которых они предназначены, однако все то, что можно выполнить, комбинируя эти средства, можно реализовать в одной Perl-программе, изучив только *один* язык. Но возможности Perl не ограничиваются только задачами администрирования. Подключаемые пакеты и модули позволяют легко и быстро решать и другие задачи, для которых, возможно, пришлось бы использовать язык программирования C. Начиная с версии 5.0, язык Perl поддерживает технологию объектно-ориентированного программирования, причем пакеты и модули можно оформить в виде объектов и использовать без знания содержащегося в них кода (хотя придется изучить большое количество объектных моделей со своими свойствами и методами).

Perl — это язык, на котором программист может *делать* свою работу, причем для выполнения одной и той же задачи Perl предлагает несколько средств ее реализации. Одни из них более сложны, другие — менее. Разработчик может выбрать то, которое ему более понятно и которое ему проще применить, не тратя времени на изучение более сложных возможностей.
В этом заключается простота использования Perl, которая позволяет применять его как для реализации одноразовых утилит, так и для создания сложных, часто используемых приложений.

Perl является прямолинейным языком, а это означает, что простые программы не надо оформлять в виде головных процедур `main`, как это принято в большинстве процедурных языков программирования, или в форме клас-

са, как принято в объектно-ориентированных языках программирования, т. е. не надо тратить время на дополнительное форматирование исходного текста программы, а просто начинать писать операторы Perl, которые будут немедленно обрабатываться интерпретатором. Именно в этом заключена эффективность языка программирования Perl.

Четвертое преимущество использования Perl связана с его дополнительными возможностями, позволяющими выполнять не только традиционные задачи администрирования UNIX и обработки текстовых файлов.

И здесь, в первую очередь, следует обратить внимание на простое включение в Perl-программу вызовов библиотечных процедур языка C, что позволяет использовать огромное количество кода, написанного для этого популярного языка. В поставку Perl входят утилиты, конвертирующие заголовки библиотек C в соответствующие эквиваленты языка Perl. Конвертирование осуществляется с помощью XS-интерфейса, который представляет собой простой программный интерфейс, преобразующий среду вызова функций C в среду вызова подпрограмм Perl. Последующий вызов функций C ничем не отличается от вызова подпрограмм самого Perl. Более того, программы Perl версии 5.0 легко интегрируются в приложения C и C++ через интерфейс, реализованный в наборе функций `perl_call_*`.

Для работы с базами данных можно самому написать соответствующее приложение на языке C, а можно воспользоваться свободно распространяемыми модулями дополнительных расширений возможностей Perl, включающих работу с многочисленными популярными системами управления базами данных: Oracle, Ingres, Informix, Interbase, Postgre, Sybase 4 и др.

Способность Perl работать с сокетами TCP/IP сделала его популярным для реализации информационных систем взаимодействия с сетевыми серверами любых типов, использующих сокеты в качестве механизма обмена информацией. Именно эта возможность в сочетании с использованием Perl для создания CGI-сценариев послужила широкому распространению языка на других многочисленных платформах.

И в завершение перечисления достоинств Perl обратим внимание читателя на **пятое** преимущество его использования: так как изначально этот язык являлся свободно распространяемым, то вся наработанная документация также доступна совершенно бесплатно, а так как Perl, как язык сценариев очень популярен, то в Internet находится море документации по его применению для решения разнообразных задач.

(Некоторые адреса можно найти в главе 16.)

1.3. Области применения Perl

Наиболее широко Perl используется для разработки инструментов системного администрирования, однако в последнее время он получил огромную популярность в области разработки Internet-приложений: CGI-сценарии, системы автоматической обработки электронной почты и поддержки узлов Web. В этом параграфе мы кратко охарактеризуем возможности Perl в каждой из указанных областей.

Системная поддержка UNIX

Как отмечалось ранее, именно задача соединения в одном языке программирования возможностей различных средств системного администрирования UNIX и послужила толчком к разработке и созданию языка Perl. Он и разрабатывался таким образом, чтобы оптимизировать решение именно этих задач, не прибегая к другим инструментам. На настоящий момент язык Perl является основным средством администрирования UNIX, который может выполнять работу нескольких других традиционных средств администрирования. Именно эта его универсальность и способствовала его широкому распространению среди системных администраторов и программистов UNIX, тем более, что он решает задачи обычно быстрее, чем другие аналогичные средства.

CGI-сценарии

Одной из первых, но продолжающей и по настоящее время широко применяться в Интернете технологией реализации динамических эффектов является технология CGI-сценариев, суть которой заключается в обработке информации, получаемой от пользователя, которую он вводит в поля формы страницы HTML, просматриваемой с помощью программы-обозревателя Internet. Информация из полей формы пересылается на сервер с помощью протокола HTTP либо в заголовке, либо в теле запроса и обрабатывается сценарием, который после анализа полученных данных выполняет определенные действия и формирует ответ в виде новой страницы HTML, отсылаемой обратно клиенту. Сценарий может быть написан, собственно говоря, на любом языке программирования, имеющем доступ к так называемым переменным среды, но сценарии Perl получили наибольшее распространение из-за легкости создания и оптимизационных возможностей языка Perl при обработке текстовых файлов. В Internet можно найти буквально тысячи примеров динамического CGI-программирования на Perl.

Его большая популярность для реализации подобных задач на UNIX-серверах Internet привела к тому, что разработчики серверов Internet, рабо-

тающих в других операционных системах, стали включать возможность подключения сценариев Perl в свои системы. В настоящее время их можно использовать и на сервере Internet Information Server фирмы Microsoft для операционных систем семейства Windows, и на серверах Apache, NCSA и Netscape для операционной системы UNIX.

Обработка почты

Другая популярная область применения Perl — автоматическая обработка электронной почты Internet. Сценарии Perl можно использовать для фильтрации почты на основе адреса или содержимого, автоматического создания списков рассылки и для решения многих других задач. Одной из наиболее популярных программ для работы с электронной почтой является программа Majordomo, полностью реализованная средствами Perl.

Возможности Perl в этой области огромны и ограничиваются только фантазией разработчика. Можно, например, написать сценарий, который обрабатывает входящую почту и добавляет сообщения на заранее созданную страницу новостей, сортируя их по соответствующим тематикам, что позволяет быстро просматривать почту, не тратя время на чтение каждой полученной корреспонденции. По прошествии определенного времени сообщения удаляются со страницы.

Поддержка узлов Web

Узел Web — это ничто иное, как структурированное хранилище страниц HTML, которые являются обычными текстовыми файлами в определенном специальном формате, понимаемом программами просмотра их содержимого. Perl оптимизирован для обработки большого количества текстовых файлов, — поэтому его использование для анализа и автоматического изменения содержимого узла Web само собой вытекает из тех задач, для решения которых он специально и создавался. Его, например, можно использовать для решения задачи проверки правильности перекрестных ссылок на страницах узла Web, как, впрочем, и для проверки правильности ссылок на другие узлы (правда, здесь придется воспользоваться его сетевыми возможностями работы с сокетами).

Его возможности записи и чтения в/из сокетов позволяют использовать сценарии Perl для взаимодействия с другими узлами и получения информации на основе протокола HTTP. Следует отметить, что существуют даже серверы, написанные на Perl. Как упоминалось ранее, именно эти возможности Perl можно использовать для удаления со страниц HTML узла Web ссылок на несуществующие другие узлы.

Perl может работать и с протоколом FTP. Это позволяет автоматизировать получение файлов с других узлов, а в сочетании с его возможностями обработки текстовых файлов позволяет создавать сложные информационные системы.

* * *

В этой главе мы попытались кратко охарактеризовать сам язык Perl, очертить основные области его применения и привлечь внимание читателя к его дальнейшему изучению и внедрению в собственную практику. В конечном счете только сам программист решает, нужен ему соответствующий язык или нет. Мы думаем, что наш уважаемый читатель уже сделал свой выбор и надеемся, что он не покинет нас до самой последней страницы книги.

Вопросы для самоконтроля

1. Назовите полное наименование языка Perl.
2. Что послужило толчком для разработки и создания Perl?
3. Каково назначение Perl-программы?
4. В чем заключаются преимущества и недостатки интерпретируемых языков?
5. Перечислите основные достоинства языка Perl.
6. Перечислите области применения Perl.

Глава 2

Структура программы

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.

2.1. 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 Perl-программы
20
21 .
22
23 format NAME_FORMAT_BOTTOM= # 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 program1.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 program1.pl
```

и запустить на выполнение из командной строки:

```
./program1.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 назы-

вается 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 `<>` 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 отобра-

жаться 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.

2.2. 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 `my` 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 объяв-

ления 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.

2.3. 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 правиль-

ные 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.

Пример 2.4. Литералы языка Perl

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