Краткое руководство по языку программирования С

Автор: Моргунов Евгений Павлович (emorgunov@)mail.ru

Версия: 0.2

10 февраля 2014 г.

1. Настоящее руководство не является учебником по программированию на языке С или по изучению основ программирования вообще. Руководство предназначено для тех, кто уже владеет каким-либо другим языком программирования, знает основные приемы программирования и терминологию (переменная, массив, оператор, цикл, функция, процедура и т. д.). Оно призвано дать лишь самое первое и самое общее представление о языке С. А затем нужно изучать учебники и техническую документацию по программированию на этом языке. Ряд книг по языку С приводится в конце этого текста.

2. Руководство представлено в виде исходных текстов программ, при этом какого-либо теоретического курса не предлагается. Однако в каждой программе присутствует большое количество комментариев, которые имеют учебный характер, эти комментарии следует не просто просматривать, а вдумчиво изучать.

3. Программы были созданы в среде операционной системы (ОС) Linux Debian 7.1.0, компилятор GCC версии 4.7.2. Компиляция и выполнение программ были протестированы в ОС FreeBSD 9.2 и Microsoft Windows 7. В ОС Windows в качестве среды программирования использовалась интегрированная среда разработки (IDE) CodeBlocks (http://www.codeblocks.org) версии 12.11 с компилятором GCC версии 4.7.1. В ОС Windows возможно также использование среды MinGW (http://www.mingw.org) и без какой-либо IDE.

Предполагается, что в ОС Unix (Linux, FreeBSD) компиляция этих программ будет выполняться из командной строки. В ОС Microsoft Windows также можно использовать командную строку в случае использования среды MinGW.

При использовании какой-нибудь IDE (например, CodeBlocks) компиляция выполняется не из командной строки, поэтому дополнительные параметры компилятора или компоновщика нужно задавать, используя соответствующие разделы меню IDE. Например, в CodeBlocks это будет меню Settings->Compiler.

Для запуска программ в ОС Windows удобно использовать файловый менеджер FAR. В нем нужно установить шрифт Lucida Console, а также назначить кодовую страницу 1251 с помощью команды

```
chcp 1251
```

Поскольку в разных ОС используются различные кодировки символов, то нужно использовать исходные тексты из соответствующего подкаталога:

UTF8 – для Debian, KOI8-R – для FreeBSD, CP1251 – для Windows. В подкаталоге CP866 находятся исходные тексты в так называемой альтернативной кодировке. Ее также можно использовать в среде Windows. Все преобразования кодировок были выполнены с помощью скрипта recode.sh.

4. Поскольку различные операционные системы имеют свои особенности, то не все представленные программы будут работать совершенно одинаково в ОС Debian, FreeBSD и Windows. В ряде случаев для учета особенностей конкретной ОС в исходных текстах программ использовались директивы препроцессора (например, #ifdef и другие). Если возникают непредвиденные ошибки при компиляции программы, то нужно проверить в исходном тексте наличие подобных директив, которые позволяют организовать так называемую условную компиляцию. В текстах программ даются указания о том, какие параметры нужно указать компилятору в таких случаях.

5. После компиляции и первого выполнения каждой программы следует выполнить задания, приведенные далее в этом тексте. Выполнение каждого задания предполагает, что вы должны внести указанное изменение в программу, сохранить измененный исходный текст, затем перекомпилировать программу, запустить ее и посмотреть, что изменилось в ее поведении (например, при выводе данных на дисплей). При внесении некоторых из предлагаемых изменений программа может даже не скомпилироваться. Так и должно быть. В таком случае нужно внимательно изучить сообщения компилятора, чтобы понимать причины этих сообщений об ошибках. Кроме сообщений об ошибках (errors) компилятор может выдавать также предупреждения (warnings). Они не препятствуют созданию исполняемого файла. Однако при его исполнении возможны ошибки в работе программы. Поэтому нужно стремиться к тому, чтобы программа компилировалась даже без предупреждений.

6. При выполнении каждого задания обязательно берите оригинальный текст каждой программы, т. к. в противном случае изменения, предлагаемые в двух различных заданиях и внесенные в текст программы одновременно, могут вступить в противоречие и исказить суть предлагаемых заданий.

Программа 1

Команда компиляции:

```
gcc -o prg1 prg1.c
```

Получаем исполняемый файл с именем prg1 (в ОС Windows это будет prg1.exe). Если параметр компилятора -o не использовать, то исполняемый файл получит по умолчанию имя a.out (в ОС Windows – a.exe).

Задания

1. Удалите символы “\n” из функции printf().

2. Удалите число 0 из оператора return.

3. Закомментируйте включение файла <stdio.h>. Для этого используйте символы /* */ или //

4. В строке компиляции уберите параметр -o prg1. Команда станет такой:

```
gcc prg1.c
```

Посмотрите, каким станет имя исполняемого файла.

Программа 2

Команда компиляции:

```
gcc -o prg2 prg2.c
```

Получаем исполняемый файл с именем prg2 (или prg2.exe в ОС Windows).

Задания

1. Уберите символ &, стоящий перед именем одной из переменных в вызовах функции scanf().

2. Замените спецификацию %.2f в вызове функции scanf() для числа с плавающей запятой на %f.

3. При выполнении оригинального (правильного) варианта программы попробуйте внести следующие изменения в процесс ввода данных с клавиатуры:

– вместо одного символа введите два символа;

– вместо целого числа введите несколько символов;

– вместо целого числа введите число с плавающей запятой;

– при вводе числа с плавающей запятой введите запятую вместо точки;

– при вводе имени введите еще и фамилию (через пробел).

4. При выполнении оригинального (правильного) варианта программы в ответ на первое предложение для ввода данных введите в одну строку следующее (через пробел):

a 21 4.75 Иван

5. Введите те же самые данные, но без пробелов между ними. В чем различия при выводе?

Программа 3

Команда компиляции:

```
gcc –o prg3 prg3.c
```

Получаем исполняемый файл с именем prg3 (или prg3.exe в ОС Windows).

Задания

1. В ответ на запрос вашего возраста введите не целое число, а число с плавающей запятой.
2. Закомментируйте первую пару вызовов функции scanf():

```
scanf( "%*[^\n]" );
scanf( "%*c" );
```

3. Закомментируйте вторую пару вызовов функции scanf():

```
scanf( "%*[^\n]" );
scanf( "%*c" );
```

а сразу после них вставьте следующие строки:

```
scanf( "%c", &letter );
printf( "В буфере еще находился символ с ASCII-кодом %d\n", letter );
```

4. Выполните задания 1 и 2 совместно.
5. В конструкции switch закомментируйте оператор break в ветви case ‘A’. Запустив затем программу, введите латинскую букву ‘a’ либо ‘A’. Для чего предназначен оператор break?

Программа 4

Команда компиляции для ОС Debian:

```
gcc –DUTF8 –o prg4 prg4.c
```

Команда компиляции для ОС FreeBSD и Windows:

```
gcc –o prg4 prg4.c
```

Получаем исполняемый файл с именем prg4 (или prg4.exe в ОС Windows).

Задания

1. При выводе значений элементов одномерного массива выведите также и индексы элементов.

2. Сделайте то же самое и для двухмерного массива.

3. При выводе возраста в словесной форме добавьте слово «лет», «год» или «года» в зависимости от того, какое число единиц в конкретном введенном значении.

4. Реализуйте этот же механизм для вывода в словесной форме трехзначных чисел.

5. С помощью следующей команды можно получить результат работы препроцессора (в ОС FreeBSD и Windows не давайте параметр –DUTF8):

```
gcc –E –DUTF8 –o prg4.pre prg4.c
```

При этом исполняемый файл не создается. Найдите в файле prg4.pre тот фрагмент исходного кода, который был включен в результате выполнения условной компиляции. Суффикс .pre не является обязательным, а был выбран автором произвольно (pre – preprocessor).

Программа 5

Команда компиляции:

```
gcc –o prg5 prg5.c
```

Получаем исполняемый файл с именем prg5 (или prg5.exe в ОС Windows).

Задания

1. Используя операцию sizeof и функцию printf(), посмотрите, сколько байтов занимают в памяти переменные типов int, char, float, double, а также указатели на эти типы.

2. При выводе элементов массива выводите их не все подряд, а через один.

3. Используя указатели, выведите элементы массива в обратном порядке. Можно использовать операцию декремента в условии цикла for.

4. Добавьте в команду компиляции параметр -Wall, заставляющий компилятор быть более требовательным и выводить (почти) все предупреждения (warnings):

```
gcc –Wall –o prg5 prg5.c
```

Попытайтесь добиться того, чтобы эти предупреждения исчезли. В том случае, когда речь идет о спецификаторе формата, воспользуйтесь приведением типа (char *).

Программа 6

Команда компиляции для Debian и FreeBSD:

```
gcc –DUNIX –o prg6 prg6.c
```

Команда компиляции для Windows:

```
gcc –o prg6 prg6.c
```

Получаем исполняемый файл с именем prg6 (или prg6.exe в ОС Windows).

Задания

1. Добавьте к данным, запрашиваемым у пользователя, еще запрос адреса.
2. Создайте в текстовом редакторе два текстовых файла, например, списки студентов двух групп (не нужно вводить много записей, достаточно по 2–3 записи). Измените программу таким образом, чтобы она могла открыть эти два файла и вывести их содержимое в третий файл: сначала содержимое первого файла, а затем второго.
3. Измените формат записи в файл таким образом, чтобы вся информация о персоне размещалась на одной строке, при этом не сохраняйте в файле служебную информацию (названия полей «Фамилия», «Имя» и т. д.).

Программа hello_world

Команды компиляции:

Сначала компилируем все три модуля по отдельности, в результате получим три объектных модуля: hello_world.o, hello.o и world.o.

```
gcc –c hello_world.c
gcc –c hello.c
gcc –c world.c
```

А затем объединяем их все в один исполняемый файл.

```
gcc –o hello_world hello_world.o hello.o world.o
```

Получаем исполняемый файл с именем hello_world (или hello_world.exe в ОС Windows).

Задания

1. Добавьте еще один модуль, который делал бы что-нибудь полезное.

Программа 7

Команда компиляции:

```
gcc –o prg7 prg7.c
```

Получаем исполняемый файл с именем prg7 (или prg7.exe в ОС Windows).

Задания

1. Переименуйте исполняемый файл и запустите программу без параметров. Вы увидите, что в сообщении, выводимом программой, используется это новое имя.

2. Передайте программе два параметра:

```
prg7 first second
```

Программа сообщит, что ей переданы два параметра.

Измените командную строку таким образом:

```
prg7 “first parameter” “second parameter”
```

Сколько параметров теперь получила программа? Уберите кавычки – что изменилось?

3. Передайте программе три параметра, состоящие из одного символа каждый:

```
prg7 a b c
```

Изучите выведенную на экран информацию об адресах параметров в памяти. Затем передайте программе три параметра, состоящие из двух символов каждый. Что изменилось в значениях адресов? Есть ли закономерность? И наконец, передайте в качестве параметров три пустые строки:

```
prg7 "" "" ""
```

Что изменилось?

Программа 8

Команда компиляции:

```
gcc –o prg8 prg8.c
```

Получаем исполняемый файл с именем prg8 (или prg8.exe в ОС Windows).

Задания

1. Запустите программу, введя в командной строке ее имя (в ОС Windows точка и косая черта перед именем программы не нужны)

```
./prg8
```

Теперь вводите какой-нибудь текст с клавиатуры и нажимайте Enter для завершения ввода каждой строки. Вы видите, что выводится копия введенного вами текста. Для завершения ввода нажмите Ctrl-D (в системе Windows нажмите Ctrl-Z, а затем Enter).

2. Запустите программу, указав ей имена файлов для переадресации ввода и вывода

```
prg8 < prg8.c > prg8_new.c
```

3. Измените программу так, чтобы она копировала не все символы, а лишь определенное их подмножество: например, все символы, кроме русской буквы «а». Еще один способ разнообразить работу этой программы – заставить ее заменять одни символы другими, например, заменять каждый символ следующим по алфавиту (это будет, пусть и наивный, но уже способ шифрования текстовых файлов). При этом заменять символы новой строки ‘\n’ не следует.

Программа 9

Команда компиляции:

```
gcc –o prg9 prg9.c
```

Получаем исполняемый файл с именем prg9 (или prg9.exe в ОС Windows).

Задания

1. При вводе строк, которые будут сравниваться, попробуйте вводить одинаковые слова, затем слова разной длины, но имеющие совпадающие первые несколько букв. Затем введите слова, уже первые буквы которых различаются. Экспериментируйте.

2. При изучении функции sprintf() убедитесь, что процедура округления вводимого значения зарплаты до двух знаков после запятой выполняется правильно. Для этого введите следующие значения зарплаты (естественно, не за один раз, а выполняя программу многократно): 123.451, 123.456, 999.999.

3. Добавьте еще какой-нибудь элемент в дополнение к имени, фамилии и зарплате, чтобы он также был включен в единую символьную строку с помощью функции sprintf().

Программа 10

Команда компиляции:

```
gcc –o prg10 prg10.c
```

Получаем исполняемый файл с именем prg10 (или prg10.exe в ОС Windows).

Задания

1. Распределите память для массива псевдослучайных чисел типа float (с плавающей запятой) и заполните его числами в диапазоне от 0 до 1. Используйте константу RAND_MAX. Не забудьте о том, что функция rand() возвращает значение целого типа (int), а при делении одного целого числа на другое (речь о целочисленной константе RAND_MAX) дробная часть просто отбрасывается. Поэтому при вызове функции rand() используйте приведение типа

```
numbers[ i ] = ( float )rand() / RAND_MAX;
```

Попробуйте и вариант без приведения типа:

```
numbers[ i ] = rand() / RAND_MAX;
```

При выводе чисел на дисплей используйте спецификацию формата %.5f, чтобы выводить 5 знаков после запятой.

2. Закомментируйте (т. е. поставьте в начале строки символы //) вызов функции srand() и запустите программу несколько раз. Какую закономерность в выводимых последовательностях чисел вы видите?

3. Добавьте в команду компиляции параметр -Wall, заставляющий компилятор выводить (почти) все предупреждения (warnings):

```
gcc –Wall –o prg10 prg10.c
```

Будет выведено предупреждение насчет неявного объявления функции time. Чтобы избавиться от предупреждения, с помощью команды

```
man 2 time
```

откройте справочное руководство по этой функции (точнее, это – системный вызов). В руководстве указано, что для использования функции time() нужно включить в программу директиву #include <time.h>. Так и поступите, а затем перекомпилируйте программу, чтобы это предупреждение исчезло.

Программа 11

Команда компиляции:

```
gcc –o prg11 prg11.c
```

Получаем исполняемый файл с именем prg11 (или prg11.exe в ОС Windows).

Задания

1. Совместите операции распределения памяти для массивов второго уровня с заполнением этих массивов случайными числами.

2. При выводе содержимого двухмерного массива уберите значение ширины поля из спецификатора формата функции printf(). Как изменилась картина?

3. Добавьте в команду компиляции параметр -Wall, заставляющий компилятор выводить (почти) все предупреждения (warnings):

```
gcc –Wall –o prg11 prg11.c
```

Будет выведено предупреждение насчет неявного объявления функции time. Избавьтесь от этого предупреждения, как в программе prg10.c.

Программа 12

Команда компиляции:

```
gcc –o prg12 prg12.c
```

Получаем исполняемый файл с именем prg12 (или prg12.exe в ОС Windows).

Задания

1. Добавьте в структуру person еще одно поле, например, год рождения (представить его можно полем типа int). Измените в соответствии с новой структурой структуры person (вот такой каламбур! а что делать?…) все операторы, которые требуют изменения, или добавьте новые операторы в программу.

Программа 13

Команда компиляции для Debian:

```
gcc –DWIDE –o prg13 prg13.c
```

Команда компиляции для FreeBSD и Windows:

```
gcc –o prg13 prg13.c
```

Получаем исполняемый файл с именем prg13 (или prg13.exe в ОС Windows).

Задания

1. Измените программу так, чтобы обход массива структур производился с конца массива, а не с начала.

2. Попробуйте при вводе данных для менеджера ввести не только те данные, которые требуются, но и, после пробела, еще какие-то (в этой же строке). При ответе на запрос о продолжении ввода данных введите не одну букву, а две или более. Пробуйте вводить не только допустимые варианты «д» и «н», но и какие-то другие. Внимательно наблюдайте за поведением программы, за тем, что она выводит на экран.

3. Закомментируйте два фрагмента

```
while ( getchar() != '\n' )
  continue;
```

Проделайте все те же «игры» с программой, что и в п. 2. Что изменилось в поведении программы?

4. Добавьте в команду компиляции параметр -Wall, заставляющий компилятор выводить (почти) все предупреждения (warnings):

```
gcc –Wall –o prg13 prg13.c
```

Будет выведено предупреждение насчет неиспользуемых переменных. Избавьтесь от этого предупреждения.

Программа 14

Команда компиляции для Debian:

```
gcc –DWIDE –o prg14 prg14.c
```

Команда компиляции для FreeBSD и Windows:

```
gcc –o prg14 prg14.c
```

Получаем исполняемый файл с именем prg14 (или prg14.exe в ОС Windows).

Задания

1. Откройте файл students.db в текстовом редакторе, например, joe или в редакторе файлового менеджера FAR или deco, или Midnight Commander. Вы увидите, что не все символы, если так можно, выразиться, поддаются чтению невооруженным глазом. Текстовые данные (имя и фамилию) найти в этом файле можно, но числовые данные – нет, потому что они хранятся в оригинальном виде, а не в виде символов. Например, число 175 хранится не в виде трех символов «1», «2» и «3», а в виде четырехбайтового целого числа. В данном случае имеет место несколько больший расход дисковой памяти, чем это было бы при хранении числа в виде текстовой строки, но при работе с большими числами двоичная форма хранения чисел уже даст экономию дисковой памяти. Но, может быть, даже важнее то, что при записи чисел типа float (с плавающей запятой) в файл в двоичном виде не происходит потери точности, которая почти неизбежна при преобразованиях из двоичной формы в строковую.

2. Введите две–три записи в файл – они будут выведены на экран. После завершения работы программы запустите ее снова и добавьте записи в файл – теперь на экран будут выведены не только старые записи, но и только что добавленные.

3. В программе используется переменная count, однако никакой видимой пользы от ее присутствия нет. Придумайте, как можно распорядиться этой переменной более разумно.

Программа 15

Команда компиляции:

```
gcc –o prg15 prg15.c
```

Получаем исполняемый файл с именем prg15 (или prg15.exe в ОС Windows).

Задания

1. Вводите различные числа в ответ на запрос программы. Начните с ввода совсем небольших значений (1, 2 и т. д.). Посмотрите, какими комбинациями битов они представлены. Учтите, что многобайтовые числовые типы данных представлены так: младший байт имеет меньший адрес. Если провести аналогию с десятичными числами, которые мы пишем на бумаге, то будет так: например, в числе 654321 младший разряд равен 1, и он – на письме (!) – располагается справа, т.е. в конце письменного представления числа. В памяти же компьютера этот разряд будет располагаться ближе к началу отсчета адресов, т. е. в начале группы ячеек, хранящих это число. Это упрощенная аналогия, поскольку в данном числе в младшем байте разместится не только самый младший разряд 1, но и еще ряд разрядов. Однако, вводя в ответ на приглашение программы сначала небольшие числа, а затем все более крупные, можно увидеть закономерность в той конфигурации нулей и единиц, которые соответствуют битам каждого числа.

Программа 16

Команда компиляции:

```
gcc –o prg16 prg16.c
```

Получаем исполняемый файл с именем prg16 (или prg16.exe в ОС Windows).

Задания

1. Раскомментируйте закомментированные операторы и скомпилируйте программу. Посмотрите, сообщения компилятора будут различаться в зави-

симости от того, что делает данный недопустимый оператор: присваивает ли он новое значение константе или пытается осуществить запись нового значения в неизменяемую область памяти.

Программа 17

Команда компиляции:

```
gcc –o prg17 prg17.c
```

Получаем исполняемый файл с именем prg17 (или prg17.exe в ОС Windows).

Задания

1. Измените программу так, чтобы параметром функций get_word_a(), get_word_b(), get_word_c() был не указатель на строку символов, а указатель на еще одну функцию.

Программа 18

Команда компиляции:

```
gcc –o prg18 prg18.c
```

Получаем исполняемый файл с именем prg18 (или prg18.exe в ОС Windows).

ПРИМЕЧАНИЕ. Эта программа компилируется и работает в среде ОС Debian. Для ее компиляции в ОС FreeBSD и Windows необходимо принять дополнительные меры.

Задания

1. Закомментируйте блок текста, в котором написан вызов функции setlocale() и скомпилируйте программу. Запустите ее. Что получается?

2. Вводя различные символы в ответ на приглашения программы, внимательно изучайте байтовое представление этих символов и символьных строк. Не забывайте, что в широких символах младший байт имеет меньший адрес, а старший байт – больший. Поэтому байты выводятся на экран, грубо говоря, задом наперед.

3. Скомпилируйте программу с ключом компилятора –DWSCANF (в тексте программы найдите директиву препроцессора, которая использует этот параметр). Снова выполните программу и поэкспериментируйте с ней.

Программа bsplit

Команда компиляции:

```
gcc –o bsplit bsplit.c
```

Получаем исполняемый файл с именем bsplit (или bsplit.exe в ОС Windows).

ПРИМЕЧАНИЕ. Эта программа – упрощенный аналог программы split, описание которой можно посмотреть с помощью команды

```
man split
```

Задания

1. Изучите исходный текст программы, в ней используется много функций языка С для работы с файлами.

2. Попробуйте использовать программу для разделения текстовых файлов на части. Например, от одного файла «отрежьте» начало, от другого – конец, а из третьего вычлените среднюю часть. При этом используйте измерение объема этих фрагментов как в байтах, так и в строках.

Программа students

В состав этой программы входят три модуля и заголовочный файл:

```
students.c
groups.c
lists.c
students.h
```

В данном случае для компиляции программы используется утилита make, которая выполняет команды, содержащиеся в так называемом make-файле. Его имя по умолчанию – Makefile. Он находится среди остальных файлов, предлагаемых вашему вниманию. В зависимости от операционной системы, в этом файле нужно оставить только одну строку с переменной CPARAMS, а две другие такие строки нужно закомментировать. При использовании IDE, например, CodeBlocks, нужно включить все эти исходные файлы в проект. А для компиляции в командной строке нужно выполнить команду

```
make
```

а в среде MinGW –

```
mingw32-make
```

При компиляции выполняются следующие команды:

```
gcc -g -c students.c -o students.o
gcc -g -c groups.c -o groups.o
gcc -g -c lists.c -o lists.o
gcc -o students students.o groups.o lists.o
```

Вы можете выполнять их и вручную, но использование утилиты make имеет целый ряд преимуществ.

Получаем исполняемый файл с именем students (или students.exe в ОС Windows).

Задания

1. Ознакомьтесь с правилами использования утилиты make. Для этого можно обратиться к следующему учебному пособию (оно выложено на сайте www.morgunov.org):

Моргунов, Е. П. Технологии разработки программ на основе инструментария с открытым исходным кодом. Вводный курс [Текст] : учеб. пособие / Е. П. Моргунов, О. Н. Моргунова, В. В. Тынченко ; НИИ СУВПТ. – Красноярск, 2006. – 148 с. – ISBN 5-98027-046-9.

2. Изучите исходные тексты этой программы. Подумайте, как можно улучшить программу. Например, при вводе данных о студентах можно организовать проверку введенного пользователем шифра группы на предмет наличия группы с указанным шифром в файле groups.db. Еще одна идея: при удалении описания группы из файла groups.db необходимо удалять также из файла students.db все записи о студентах, в поле «Шифр группы» которых записан шифр удаляемой группы. И еще одна идея: организовать проверки на дублирование вводимых данных.

3. Скомпилируйте программу с параметром -Wall. Можно добавить его в переменную CPARAMS в make-файле. Добейтесь того, чтобы не было предупреждений компилятора.

Рекомендуемая литература

1. Керниган, Б. Язык программирования С [Текст] : 2-е изд. : пер. с англ. / Брайан Керниган, Деннис Ритчи. – М. : Вильямс, 2006. – 304 с. : ил. – ISBN 5-8459-0891-4.

2. Прата, С. Язык программирования С. Лекции и упражнения [Текст] : 5-е изд. : пер. с англ. / Стивен Прата. – М. : Вильямс, 2006. – 960 с. : ил. – ISBN 5-8459-0986-4.

3. Хэзфилд, Р. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений [Текст] : энциклопедия программиста : пер. с англ. / Ричард Хэзфилд, Лоуренс Кирби [и др.]. – К. : ДиаСофт, 2001. – 736 с. – ISBN 966-7393-82-8.

4. Шилдт, Г. Полный справочник по С [Текст] : 4-е изд. : пер. с англ. / Герберт Шилдт. – М. : Вильямс, 2002. – 704 с. : ил. – ISBN 5-8459-0226-6.