
ДМК ПРЕСС
издательство
C для программистов
С введением в C11
ПОЛ ДЕЙТЕЛ • ХАРВИ ДЕЙТЕЛ

УДК 004.438Си(075.8)
ББК 32.973-018.1я73-1
Д27

Дейтел П., Дейтел Х.
Д27 С для программистов с введением в С11 / пер. с анг. А. Киселева. – М.: ДМК Пресс, 2016. – 544 с.: ил.

ISBN 978-5-97060-205-8

В книге рассказывается о языке С и стандартной библиотеке С, следуя девизу компании Deitel: «обучение на действующих примерах». Понятия представляются в контексте тщательно протестированных программ, с выделением синтаксиса, демонстрацией вывода программ и подробного их обсуждения. Приводится примерно 5 000 строк кода на языке С и даются сотни полезных советов, которые помогут вам создавать надежные приложения.

Рассматривается создание собственных структур данных и стандартная библиотека, безопасное программирование на С; описываются особенности новой ревизии стандарта С11, в т. ч. многопоточность. Закончив чтение, вы будете иметь все знания, необходимые для создания приложений на языке С промышленного уровня.

Издание предназначено программистам, имеющим опыт работы на высокоуровневых языках.

УДК 004.438Си(075.8)
ББК 32.973-018.1я73-1

ISBN 978-0-13-346206-4 (анг.)
ISBN 978-5-97060-205-8 (рус.)

Содержание

1

Введение

В этой главе вы познакомитесь:

- с историей развития языка C;
- с назначением стандартной библиотеки языка C;
- с элементами типичной среды разработки программ на языке C;
- с небольшим игровым приложением, выполняющимся в Windows, Linux и Mac OS X;
- с коммерческими и свободно распространяемыми операционными системами для компьютеров и смартфонов, для которых можно писать приложения на языке C.

Краткое содержание

1.1 Введение

Добро пожаловать в язык C – выразительный и мощный язык программирования, который с успехом можно использовать для создания крупных программных систем. Книга «C для программистов» в этом отношении является отличным учебником. Она подчеркивает эффективность структурного подхода к разработке программного обеспечения и включает 130 примеров законченных программ с результатами их выполнения. Мы называем это «обучением на действующих примерах». Исходный код всех программ, описываемых в книге, можно получить по адресу: `www.deitel.com/books/cfp/`.

1.2 Язык программирования C

Свою родословную язык C ведет от двух языков: BCPL и B. Язык BCPL был создан в 1967 году Мартином Ричардсом (Martin Richards) как язык для создания операционных систем и компиляторов. Кен Томпсон (Ken Thompson) заимствовал из него многие особенности при создании своего языка B, и в 1970 году он использовал B для создания первых версий UNIX в Bell Laboratories.

Дальнейшим продолжением языка B стал язык C, разработанный Денисом Ритчи (Dennis Ritchie), сотрудником Bell Laboratories, в 1972 году. Первоначально язык C был широко известен как язык разработки операционной системы UNIX. Многие современные операционные системы также написаны на C и/или C++. Язык C практически независим от аппаратной архитектуры – при надлежащем подходе к проектированию он позволяет писать программы, способные выполняться на самых разных аппаратных платформах.

Высокая производительность

Язык C широко используется для создания систем, где требуется высокая производительность, таких как операционные системы, встраиваемые системы, системы реального времени и системы связи (табл. 1.1).

В конце 70-х годов прошлого столетия язык C развился в то, что теперь называют «традиционный C». Книга Кернигана и Ритчи «The C Programming Language»[1], вышедшая в 1978 году, привлекла широкое внимание к языку. Она стала самой продаваемой книгой по информатике из тех, что когда-либо выпускались.

Таблица 1.1 | Некоторые приложения на языке C, с высокими требованиями к производительности

Приложение	Описание
Операционные системы	Переносимость и производительность языка C являются одними из самых востребованных его характеристик при реализации операционных систем, таких как Linux и Microsoft Windows, а также Google Android. Операционная система OS X, выпускаемая компанией Apple, написана на языке Objective-C, который корнями уходит в язык C. Некоторые основные операционные системы для компьютеров и мобильных устройств мы обсудим в разделе 1.7
Встраиваемые системы	Кроме компьютеров общего назначения, каждый год выпускается огромное количество микропроцессоров для встраиваемых устройств. К числу встраиваемых систем относятся системы для навигационных устройств, комплексов «Умный дом», охранных устройств, смартфонов, роботов, комплексов управления дорожным движением и многие другие. Язык C является одним из самых популярных языков разработки подобных систем, от которых обычно требуются высочайшая скорость выполнения и низкое потребление памяти. Например, автомобильная антиблокировочная система должна немедленно реагировать на замедление скорости вращения колес, чтобы предотвратить их блокировку; игровые приставки должны работать очень быстро, чтобы обеспечить плавность мультипликации и быструю реакцию на действия игрока
Системы реального времени	Системы реального времени часто используются для выполнения «критически важных» приложений, имеющих очень жесткие требования ко времени отклика. Например, система управления движением воздушных судов должна постоянно отслеживать положение и скорость движения самолетов и сообщать эту информацию авиадиспетчерам без каких-либо задержек, чтобы они могли вовремя сообщить экипажу самолета о необходимости изменить курс для предотвращения столкновения
Системы связи	Системы связи должны быстро управлять движением огромных объемов информации, чтобы обеспечить своевременную доставку аудио- и видеопотоков адресатам

[1] Керниган Б., Ритчи Д. Язык программирования C. – 2-е изд., перераб. и доп. – ISBN: 978-5-8459-0891-9. – М.: Вильямс, 2008. – *Прим. перев.*

Стандартизация

Быстрое распространение поддержки языка С на различные аппаратные и программные платформы способствовало появлению многочисленных особенностей, похожих, но часто несовместимых. Эти различия были серьезной проблемой для программистов, кому требовалось создавать программы для нескольких платформ сразу. Со всей очевидностью встала проблема стандартизации С. В 1983 году под эгидой Американского национального комитета по стандартизации вычислительной техники и обработки информации (American National Standards Committee on Computers and Information Processing – X3) был создан технический комитет X3J11 с целью «выработать однозначное, аппаратно-независимое определение языка». В 1989 году стандарт, получивший название ANSI X3.159-1989, был одобрен сначала в Соединенных Штатах **национальным институтом стандартов США** (**American National Standards Institute, ANSI**), а затем и **международной организацией по стандартизации** (**International Standards Organization, ISO**). Мы называем его просто «Стандарт С». Этот стандарт был дополнен в 1999 году – он получил название INCITS/ISO/IEC 9899-1999, но чаще называется просто C99. Копию текста стандарта можно заказать в национальным институте стандартов США (`www.ansi.org`) по адресу: `webstore.ansi.org/ansidocstore`.

Новый стандарт С

Мы также познакомим вас с новым стандартом С (называется C11), который был одобрен к моменту выхода книги. Новый стандарт расширяет возможности языка С. Не все современные компиляторы С поддерживают эти новые возможности. И даже те, которые обладают такой поддержкой, реализуют лишь подмножество новых особенностей. Мы добавили в книгу дополнительные разделы, где описываются новые особенности языка, поддерживаемые наиболее распространенными компиляторами.

Поскольку язык С является аппаратно-независимым, программы, написанные на нем, часто могут выполняться в самых разных системах после небольших переделок или вообще без них.

1.3 Стандартная библиотека

Программы на языке С состоят из функций. Вы можете сами написать все функции, используемые программой, но большинство программистов предпочитает пользоваться богатой коллекцией уже имеющихся функций, входящих в состав **стандартной библиотеки языка С**. То есть обучение программированию на языке С фактически можно разделить на две части – обучение самому языку С и обучение особенностям использования функций из стандартной библиотеки. На протяжении книги мы рассмот-

рим множество этих функций. Программистам, желающим глубже изучить библиотечные функции, их реализацию и особенности использования в переносимом коде, рекомендуется обратиться к книге П. Дж. Плаугера (P. J. Plauger) «The Standard C Library». Посетите также веб-сайт с документацией по стандартной библиотеке C:

```
http://ru.cppreference.com/w/c
```

Язык C поощряет создание программ методом *конструирования из строительных блоков*. Старайтесь не изобретать колесо. Используйте уже существующие компоненты. Это называется **повторным использованием программного обеспечения**. При программировании на C обычно используются следующие строительные блоки:

- функции из стандартной библиотеки;
- функции, созданные вами;
- функции, созданные другими программистами (пользующимися вашим доверием).

Преимущество создания собственных функций – в том, что вы точно знаете, как они действуют. Исходный код функций остается доступным. Недостаток – дополнительные затраты времени и сил на проектирование, разработку и отладку новых функций.

Применение функций из стандартной библиотеки вместо своих собственных может повысить производительность программы, потому что эти функции написаны с прицелом на достижение максимальной производительности.

Применение функций из стандартной библиотеки вместо своих собственных может повысить переносимость программы, потому что эти функции написаны с прицелом на достижение максимальной переносимости.

1.4 C++ и другие C-подобные языки

Язык C++ был создан Бьерном Страуструпом (Bjarne Stroustrup) из Bell Laboratories. Своими корнями он уходит в язык C, добавляя к нему множество новых особенностей. Самой важной его особенностью является поддержка **объектно-ориентированного программирования**. **Объекты** являются основными **компонентами** повторно используемого программного обеспечения, моделирующими сущности реального мира. В табл. 1.2 перечислено несколько других C-подобных языков программирования, пользующихся большой популярностью.

Таблица 1.2 | Популярные C-подобные языки программирования

Приложение	Описание
Objective-C	Objective-C – это объектно-ориентированный C-подобный язык. Был создан в начале 1980-х и позднее приобретен компанией NeXT, которая, в свою очередь, была приобретена компанией Apple. Он стал основным языком программирования для операционной системы Mac OS X и всех устройств, действующих под управлением iOS (таких как iPod, iPhone и iPad)
Visual C#	Тремя основными объектно-ориентированными языками программирования в корпорации Microsoft являются Visual Basic, Visual C++ (основан на C++) и C# (основан на C++ и Java и разрабатывался для обеспечения интеграции приложений с веб)
Java	В 1991 году компания Sun Microsystems основала исследовательский проект, в результате которого появился C++-подобный объектно-ориентированный язык программирования Java. Основной целью Java является поддержка возможности писать программы, способные выполняться на самых разных компьютерах и микропроцессорных устройствах. Для этого даже придумали термин: «написанное один раз выполняется где угодно». Язык Java используется для создания масштабируемых приложений уровня предприятия, расширения функциональных возможностей веб-серверов (компьютеров, возвращающих содержимое для отображения в наших веб-браузерах), реализации приложений для бытовых устройств (смартфонов, телевизионных приставок и др.) и для многих других целей
PHP	PHP – объектно-ориентированный, открытый (раздел 1.7) язык сценариев, основанный на C и поддерживаемый сообществом пользователей и разработчиков – используется на многих веб-сайтах, включая Wikipedia и Facebook. PHP является платформонезависимым языком – существуют его реализации для всех основных операционных систем, таких как UNIX, Linux, Mac и Windows. PHP также поддерживает множество баз данных, включая открытую MySQL. В числе других языков с аналогичной концепцией можно назвать Perl и Python
JavaScript	Язык JavaScript был разработан в компании Netscape и получил широчайшее распространение как язык сценариев. Основная область его применения – добавление программируемости в веб-страницы, например анимационных эффектов и элементов интерактивности. Поддерживается всеми основными веб-браузерами

1.5 Типичная среда разработки приложений на языке C

Системы на языке C обычно состоят из нескольких частей: среды разработки программ, языка и стандартной библиотеки. Далее в этом разделе рассматривается типичная среда разработки на языке C, изображенная на рис. 1.1.

Обычно программы на языке C проходят шесть этапов (рис. 1.1): **редактирование**, **препроцессинг**, **компиляция**, **компоновка**, **загрузка** и **выполнение**. В этом разделе мы сосредоточимся на типичной Linux-системе, основанной на языке C.

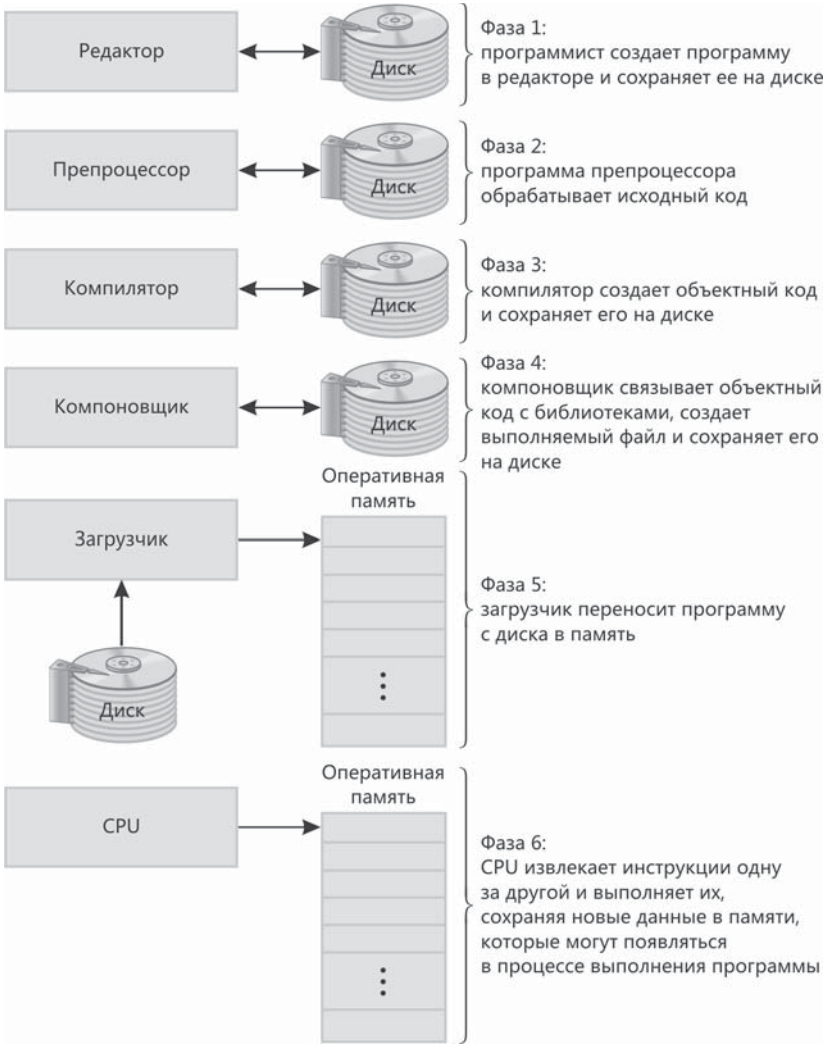

Рис. 1.1 | Типичная среда разработки приложений на языке C

1.5.1 Фаза 1: создание программы

Фаза 1 заключается в редактировании исходного кода программы. Эта фаза протекает в **программе-редакторе**. Двумя наиболее широко используемыми редакторами в Linux являются `vi` и `emacs`. Интегрированные среды

разработки программ на языке C/C++, такие как Eclipse и Microsoft Visual Studio, имеют встроенные редакторы. В течение этой фазы вы вводите исходный код программы с помощью редактора, корректируете его при необходимости, затем сохраняете на устройстве постоянного хранения, таком как жесткий диск. Файлы с исходным кодом программ на языке C должны иметь имена с расширением `.c`.

1.5.2 Фазы 2 и 3: препроцессинг и компиляция программы

В фазе 2 вы даете команду **скомпилировать** программу. Компилятор транслирует исходный код программы в машинный код (также называется объектным кодом). В системах на языке C перед фазой трансляции автоматически запускается **программа-препроцессор**. **Препроцессор языка C** выполняет специальные команды, называемые **директивами препроцессора**, определяющие операции, которые должны быть выполнены до компиляции. Обычно такими операциями являются подключение других файлов к компилируемому файлу и замена некоторых участков текста. Большинство распространенных директив препроцессора будут обсуждаться в первых главах; а в главе 13 мы более подробно рассмотрим возможности препроцессора.

В фазе 3 компилятор транслирует программу на языке C в машинный код. Если он не сможет распознать какую-либо инструкцию, генерируется **синтаксическая ошибка**, так как это считается нарушением правил языка. Чтобы помочь найти и исправить ошибочную инструкцию, компилятор выводит сообщение. Стандарт C не определяет конкретный текст сообщений об ошибках, выводимых компилятором, поэтому то, что вы увидите в своей системе, может отличаться от того, что выводится в других системах. Синтаксические ошибки также называют **ошибками компиляции**, или **ошибками времени компиляции**.

1.5.3 Фаза 4: компоновка

Следующая фаза – **компоновка**. Обычно программы на языке C содержат ссылки на функции, находящиеся где-то в другом месте, таком как стандартные библиотеки или частные библиотеки группы программистов, работающих над совместным проектом. Объектный код, производимый компилятором, чаще всего содержит «дырки», зарезервированные под отсутствующие части. **Компоновщик** связывает объектный код с кодом отсутствующих функций и генерирует **выполняемый образ** (без «дырок»). В типичной системе Linux компиляция и компоновка программ на языке C выполняются командой **`gcc`** (GNU C compiler[1]). Чтобы скомпилировать

[1] Точнее, GNU Compiler Collection – коллекция компиляторов GNU, в которой компилятор языка C – лишь один из многих. – *Прим. перев.*

и скомпоновать программу, хранящуюся в файле `welcome.c`, достаточно ввести команду

```
gcc welcome.c
```

в командной строке Linux и нажать клавишу *Enter* (или *Return*). [Имейте в виду, что команды в Linux чувствительны к регистру символов; все символы в команде `gcc` должны быть символами нижнего регистра, а регистр символов, составляющих имя файла в командной строке, должен соответствовать оригинальному имени файла.] Если программа будет скомпилирована и скомпонована без ошибок, будет создан файл `a.out`. Это – выполняемый образ программы `welcome.c`.

1.5.4 Фаза 5: загрузка

Следующая фаза – загрузка. Прежде чем программа будет запущена, ее необходимо **загрузить** в оперативную память. Эту операцию выполняет **загрузчик**, который читает выполняемый образ с диска и копирует его в оперативную память. Также загрузчик производит загрузку необходимых разделяемых библиотек.

1.5.5 Фаза 6: выполнение

Наконец, компьютер под управлением своего процессора (CPU) выполняет программу инструкция за инструкцией. Чтобы загрузить и выполнить программу в системе Linux, введите в командной строке `./a.out` и нажмите клавишу *Enter*.

1.5.6 Потоки стандартного ввода, стандартного вывода и стандартного вывода ошибок

Большинство программ на языке C вводит и/или выводит некоторые данные. Некоторые функции принимают исходные данные из **`stdin`** (**standard input stream – поток стандартного ввода**), каковым обычно является клавиатура, но поток `stdin` может быть подключен к другому потоку. Данные часто выводятся в **`stdout`** (**standard output stream – поток стандартного вывода**), каковым обычно является экран компьютера, но поток `stdout` может быть подключен к другому устройству. Когда мы говорим, что «программа вывела результаты», мы обычно подразумеваем отображение результатов на экране. Данные могут выводиться также в такие устройства, как диски и принтеры. Кроме того, существует еще **поток стандартного вывода ошибок** (**standard error stream**), который имеет имя **`stderr`**. Поток `stderr` (обычно подключенный к экрану) используется для отображения сообщений об ошибках. На практике часто используется прием перенаправления выводимых данных (то есть потока `stdout`) в устройство, отличное от экрана, а поток `stderr` оставляют подключенным к экрану, чтобы пользователь мог незамедлительно получать сообщения об ошибках.

1.6 Пробное приложение на языке C для Windows, Linux и Mac OS X

В этом разделе вы запустите и опробуете свое первое приложение на языке C. Это приложение – игра «угадай число», которая выбирает случайное число в диапазоне от 1 до 1000 и предлагает вам угадать его. Если ответ правильный – игра заканчивается. Если ответ неправильный, приложение сообщает, что ваше число больше или меньше загаданного. Ограничений на количество попыток нет, но вы должны стараться угадать число не более чем за 10 попыток. В основе этой игры лежит один интересный алгоритм – в разделе 6.8 «Поиск в массивах» вы познакомитесь с приемом поиска методом *дихотомии* (половинного деления).

Обычно подобные приложения выбирают *случайные* числа. Но в данном случае программа загадывает одну и ту же последовательность при каждом ее запуске (впрочем, такое ее поведение может зависеть от компилятора), поэтому вы можете запомнить последовательность загадываемых чисел, которая приводится ниже в данном разделе, и получить те же результаты.

Мы будем демонстрировать работу приложения в командной строке Windows, в командной оболочке Linux и в окне терминала в Mac OS X. Приложение действует совершенно одинаково на всех трех платформах. Поиграв с приложением на своей платформе, вы сможете опробовать усовершенствованную версию игры, которую можно найти в папке `randomized_version` в загружаемом архиве с примерами.

Скомпилировать и запустить программу можно в любой из множества сред разработки, таких как GNU C, Dev C++, Microsoft Visual C++, CodeLite, NetBeans, Eclipse, Xcode и др. Большинство сред разработки на языке C++ способны компилировать программы на обоих языках, C и C++.

Далее описываются шаги, выполнив которые, вы сможете запустить приложение и попытаться угадать правильное число. Элементы и функции, которые вы увидите в этом приложении, являются типичными конструкциями, с которыми вы будете знакомиться в этой книге, обучаясь программированию. Названия элементов интерфейса, отображаемых на экране, мы будем выделять рубленым шрифтом (например, **Command Prompt** (**Командная строка**)), чтобы их можно было отличить от элементов, не связанных с интерфейсом. Моноширинным шрифтом мы будем выделять имена файлов, текст, отображаемый приложением на экране, и значения, которые вводятся с клавиатуры (например, `GuessNumber` или `500`). Обратите также внимание, что **определения ключевых терминов** будут выделяться жирным шрифтом. При создании скриншотов в Windows мы изменили цвет фона окна **Command Prompt** (**Командная строка**), чтобы сделать его более читабельным. Чтобы изменить настройки цвета в приложении **Command Prompt** (**Командная строка**), запустите его, выбрав пункт меню **Start** (**Пуск**) > **All Programs**

(**Все программы**) > **Accessories** (**Стандартные**) > **Command Prompt** (**Командная строка**), щелкните правой кнопкой мыши на заголовке окна и в контекстном меню выберите пункт **Properties** (**Свойства**). В появившемся диалоге **«Command Prompt» Properties** (**Свойства «Командная строка»**) перейдите на вкладку **Colors** (**Цвета**) и выберите желаемый цвет текста и фона.

1.6.1 Запуск приложения из командной строки в Windows

1. ***Проверьте все необходимые настройки***. Обязательно прочитайте раздел «Before You Begin» на сайте `www.deitel.com/books/cfp/` и убедитесь, что без ошибок скопировали исходные тексты примеров на свой компьютер.
2. ***Перейдите в каталог с исходными текстами приложения***. Откройте окно **Command Prompt** (**Командная строка**). Перейдите в каталог с исходными текстами приложения **GuessNumber**: введите команду `cd C:\examples\ch01\GuessNumber\Windows`, затем нажмите клавишу *Enter* (рис. 1.2). Команда `cd` используется для смены каталога.

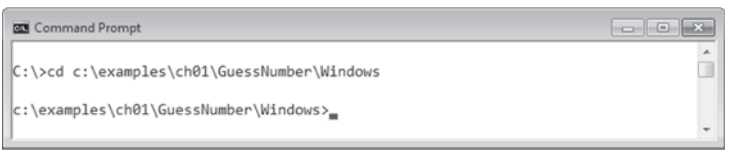

Рис. 1.2 | Откройте окно **Command Prompt** (**Командная строка**) и перейдите в каталог приложения

3. ***Запустите приложение GuessNumber***. Теперь, находясь в каталоге с исходными текстами приложения **GuessNumber**, введите команду `GuessNumber` (рис. 1.3) и нажмите клавишу *Enter*. [Обратите внимание: в действительности выполняемый файл программы имеет имя `GuessNumber.exe`, но операционная система Windows автоматически попробует добавить расширение `.exe`, если оно не указано.]

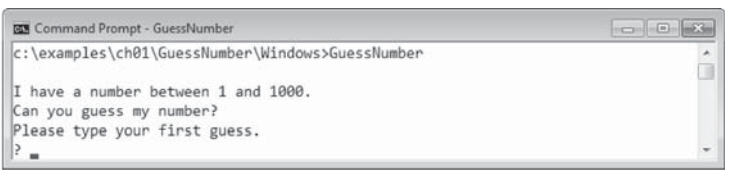

Рис. 1.3 | Запустите приложение GuessNumber

4. ***Введите первое число***. Приложение выведет приглашение к вводу `"Please type your first guess."` (Пожалуйста, введите первое число) и знак вопроса (?) в следующей строке (рис. 1.3). Введите число `500` (рис. 1.4).

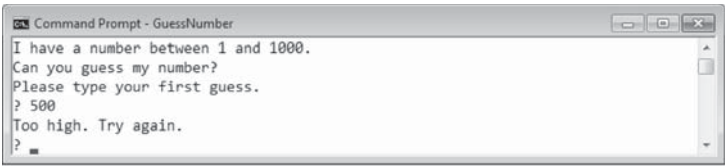

Рис. 1.4 | Введите первое число

5. ***Введите следующее число***. Приложение выведет сообщение `"Too high. Try again."` (Слишком большое, попробуйте еще), подсказывая, что введенное вами число больше загаданного. Теперь вы должны ввести число меньше предыдущего. В строке приглашения к вводу введите `250` (рис. 1.5). Приложение снова сообщит `"Too high. Try again."`, потому что введенное число также больше загаданного.

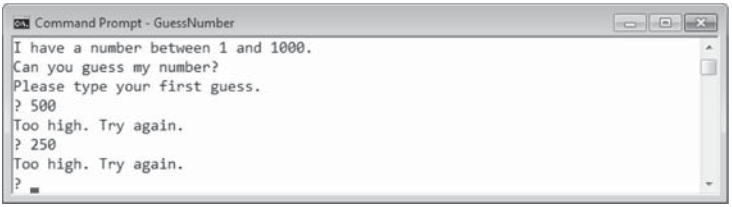

Рис. 1.5 | Введите второе число и получите ответ

6. ***Выполните дополнительные попытки***. Продолжайте игру, вводя числа, пока не угадаете загаданное. Когда это произойдет, приложение выведет `"Excellent! You guessed the number!"` (Отлично! Вы угадали число!) (рис. 1.6).
7. ***Сыграйте еще раз или завершите приложение***. Когда вы угадаете число, приложение предложит вам сыграть еще раз (рис. 1.7). Если ввести `1` в ответ на вопрос, приложение загадает новое число и выведет сообщение `"Please type your first guess."` со знаком вопроса в следующей строке (рис. 1.7), после чего вы сможете выполнить первую попытку в новом сеансе игры. Ввод числа `2` завершит приложение (рис. 1.8). Каждый раз, когда вы будете запускать приложение

заново (то есть с шага 3), оно будет загадывать те же числа и в той же последовательности.

8. ***Закройте окно Command Prompt (Командная строка).***

Рис. 1.6 | Продолжайте попытки, пока не угадаете число

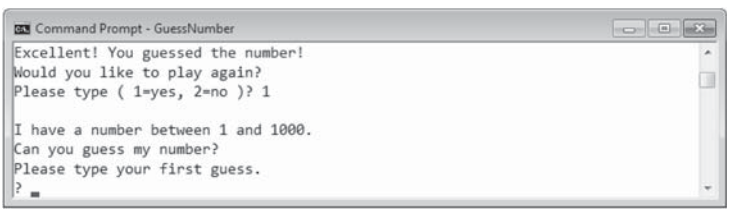

Рис. 1.7 | Сыграйте еще раз

Рис. 1.8 | Покиньте игру

1.6.2 Запуск приложения в Linux

Мы полагаем, вы уже знаете, как скопировать примеры в свой домашний каталог. На скриншотах в этом разделе мы будем выделять ввод пользователя жирным шрифтом. В нашей системе строка приглашения к вводу начинается с символа тильды (~), представляющего домашний каталог, и заканчивается знаком доллара ($). В других системах Linux строка приглашения к вводу может отличаться.

1. ***Проверьте все необходимые настройки***. Обязательно прочитайте раздел «Before You Begin» на сайте `www.deitel.com/books/cfp/` и убедитесь, что без ошибок скопировали исходные тексты примеров на свой компьютер.
2. ***Перейдите в каталог с исходными текстами приложения***. В командной оболочке Linux перейдите в каталог с исходными текстами приложения **GuessNumber** (рис. 1.9), введя команду:

```
cd examples/ch01/GuessNumber/GNU
```

Затем нажмите клавишу *Enter*. Команда `cd` используется для смены каталога.

```
~$ cd examples/ch01/GuessNumber/GNU
~/examples/ch01/GuessNumber/GNU$
```

Рис. 1.9 | Перейдите в каталог приложения

3. ***Скомпилируйте приложение GuessNumber***. Чтобы запустить приложение, его сначала необходимо скомпилировать (рис. 1.10), введя команду

```
gcc GuessNumber.c -o GuessNumber
```

Она скомпилирует исходный код и создаст выполняемый файл `GuessNumber`.

```
~/examples/ch01/GuessNumber/GNU$ gcc GuessNumber.c -o GuessNumber
~/examples/ch01/GuessNumber/GNU$
```

Рис. 1.10 | Скомпилируйте приложение GuessNumber с помощью команды gcc

4. ***Запустите приложение GuessNumber***. Чтобы запустить выполняемый файл `GuessNumber`, введите команду `./GuessNumber` и нажмите клавишу *Enter* (рис. 1.11).
5. ***Введите первое число***. Приложение выведет приглашение к вводу `"Please type your first guess."` (Пожалуйста, введите первое число) и знак вопроса (`?`) в следующей строке (рис. 1.11). Введите число `500` (рис. 1.12).