

С для программистов

С введением в С11

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 англ. 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

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

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

© 2013 by Pearson Education, Inc.
© Оформление, перевод,
ДМК Пресс, 2014

Содержание

Предисловие	15
1 Введение	23
1.1 Введение	24
1.2 Язык программирования C	24
1.3 Стандартная библиотека	26
1.4 C++ и другие C-подобные языки	27
1.5 Типичная среда разработки приложений на языке C	28
1.5.1 Фаза 1: создание программы	29
1.5.2 Фазы 2 и 3: препроцессинг и компиляция программы	30
1.5.3 Фаза 4: компоновка	30
1.5.4 Фаза 5: загрузка	31
1.5.5 Фаза 6: выполнение	31
1.5.6 Поток стандартного ввода, стандартного вывода и стандартного вывода ошибок	31
1.6 Пробное приложение на языке C для Windows, Linux и Mac OS X	32
1.6.1 Запуск приложения из командной строки в Windows	33
1.6.2 Запуск приложения в Linux	36
1.6.3 Запуск приложения в Mac OS X	39
1.7 Операционные системы	42
1.7.1 Windows – коммерческая операционная система	42
1.7.2 Linux – открытая операционная система	42
1.7.3 Apple Mac OS X, Apple iOS® для устройств iPhone®, iPad® и iPod Touch®	43
1.7.4 Google Android	44
2 Введение в программирование на C	45
2.1 Введение	46
2.2 Простая программа на C: вывод строки текста	46
2.3 Еще одна простая программа на C: сложение двух целых чисел	51
2.4 Арифметические операции в языке C	56
2.5 Принятие решений: операторы сравнения	60
2.6 Безопасное программирование на C	65

3	Управляющие инструкции: часть I	67
3.1	Введение	68
3.2	Управляющие структуры	68
3.3	Инструкция выбора if	70
3.4	Инструкция выбора if...else	71
3.5	Инструкция повторения while	75
3.6	Определение средней оценки с помощью инструкции повторения, управляемой счетчиком.....	76
3.7	Определение средней оценки с помощью инструкции повторения, управляемой сигнальным значением.....	77
3.8	Вложенные управляющие инструкции.....	81
3.9	Операторы присваивания	84
3.10	Операторы инкремента и декремента	85
3.11	Безопасное программирование на C	87
4	Управляющие инструкции: часть II	91
4.1	Введение	92
4.2	Основы повторения.....	92
4.3	Повторение со счетчиком	93
4.4	Инструкция повторения for	95
4.5	Инструкция for: замечания	98
4.6	Примеры использования инструкции for	99
4.7	Инструкция множественного выбора switch	103
4.8	Инструкция повторения do...while	110
4.9	Инструкции break и continue	112
4.10	Логические операторы	114
4.11	Путаница между операторами равенства (==) и присваивания (=)	118
4.12	Безопасное программирование на C	120
5	Функции	122
5.1	Введение	123
5.2	Модульное программирование на языке C	123
5.3	Функции из математической библиотеки	125
5.4	Функции.....	126
5.5	Определение функций.....	127
5.6	Прототипы функций: обсуждение деталей	132
5.7	Стек вызовов функций и кадры стека.....	135
5.8	Заголовочные файлы	138
5.9	Передача аргументов по значению и по ссылке	141

8 Содержание

5.10	Генератор случайных чисел.....	141
5.11	Пример: игра в кости	147
5.12	Классы хранения	151
5.13	Правила видимости	153
5.14	Рекурсия.....	157
5.15	Пример использования рекурсии: числа Фибоначчи.....	161
5.16	Рекурсия и итерации	165
5.17	Безопасное программирование на С	167

6 Массивы..... 168

6.1	Введение	169
6.2	Массивы	169
6.3	Определение массивов	171
6.4	Примеры массивов.....	171
6.5	Передача массивов функциям	185
6.6	Сортировка массивов	190
6.7	Пример: вычисление математического ожидания, медианы и моды	193
6.8	Поиск в массивах.....	197
6.9	Многомерные массивы	203
6.10	Массивы переменной длины.....	210
6.11	Безопасное программирование на С	213

7 Указатели..... 216

7.1	Введение	217
7.2	Переменные-указатели, определение и инициализация	217
7.3	Операторы указателей	219
7.4	Передача аргументов функциям по ссылке.....	221
7.5	Использование квалификатора const с указателями	224
7.5.1	Преобразование строк в верхний регистр с использованием изменяемого указателя на изменяемые данные	227
7.5.2	Вывод строки по одному символу с использованием изменяемого указателя на константные данные.....	228
7.5.3	Попытка изменить константный указатель на изменяемые данные	231
7.5.4	Попытка изменить константный указатель на константные данные	231
7.6	Пузырьковая сортировка с передачей аргументов по ссылке	232
7.7	Оператор sizeof	236
7.8	Выражения с указателями и арифметика указателей.....	238

7.9	Связь между указателями и массивами.....	242
7.10	Массивы указателей	246
7.11	Пример: тасование и раздача карт.....	247
7.12	Указатели на функции	251
7.13	Безопасное программирование на С	256
8	Символы и строки.....	258
8.1	Введение	259
8.2	Основы работы со строками и символами.....	259
8.3	Библиотека функций для работы с символами	262
8.3.1	Функции isdigit, isalpha, isalnum и isxdigit	263
8.3.2	Функции islower, isupper, tolower и toupper.....	265
8.3.3	Функции isspace, iscntrl, ispunct, isprint и isgraph	266
8.4	Функции преобразования строк.....	268
8.4.1	Функция strtod.....	268
8.4.2	Функция strtol.....	269
8.4.3	Функция strtoul.....	270
8.5	Стандартная библиотека ввода/вывода.....	271
8.5.1	Функции fgetc и putchar	271
8.5.2	Функция getchar	274
8.5.3	Функция sprintf	274
8.5.4	Функция sscanf	275
8.6	Функции для работы со строками	276
8.6.1	Функции strcpy и strncpy.....	277
8.6.2	Функции strcat и strncat	278
8.7	Функции сравнения строк.....	279
8.8	Функции поиска в строках.....	281
8.8.1	Функция strchr	282
8.8.2	Функция strcspn	282
8.8.3	Функция strpbrk	283
8.8.4	Функция strrchr	283
8.8.5	Функция strspn	284
8.8.6	Функция strstr	285
8.8.7	Функция strtok.....	285
8.9	Функции для работы с памятью	287
8.9.1	Функция memcpy.....	288
8.9.2	Функция memmove	288
8.9.3	Функция memcpy	289
8.9.4	Функция memchr	290
8.9.5	Функция memset	290

10 Содержание

8.10	Прочие функции для работы со строками	291
8.10.1	Функция <code>strerror</code>	291
8.10.2	Функция <code>strlen</code>	292
8.11	Безопасное программирование на C	292

9 Форматированный ввод/вывод 294

9.1	Введение	295
9.2	Потоки данных	295
9.3	Форматированный вывод с помощью <code>printf</code>	296
9.4	Вывод целых чисел.....	296
9.5	Вывод вещественных чисел.....	298
9.6	Вывод строк и символов	300
9.7	Прочие спецификаторы формата.....	301
9.8	Вывод с указанием ширины поля и точности.....	302
9.9	Использование флагов в строке формата функции <code>printf</code>	305
9.10	Вывод литералов и экранированных последовательностей.....	307
9.11	Чтение форматированного ввода с помощью функции <code>scanf</code>	308
9.12	Безопасное программирование на C	315

10 Структуры, объединения, перечисления и поразрядные операции 316

10.1	Введение	317
10.2	Определение структур	318
10.2.1	Структуры со ссылками на самих себя.....	318
10.2.2	Определение переменных структурных типов	319
10.2.3	Имена структур	319
10.2.4	Операции над структурами	320
10.3	Инициализация структур	321
10.4	Доступ к полям структур	321
10.5	Передача структур функциям	323
10.6	<code>typedef</code>	324
10.7	Пример: высокопроизводительная программа перемешивания и раздачи колоды карт	325
10.8	Объединения.....	327
10.8.1	Объявление объединений	328
10.8.2	Операции над объединениями.....	328
10.8.3	Инициализация объединений в объявлениях.....	328
10.8.4	Демонстрация объединений	329
10.9	Поразрядные операторы	330
10.9.1	Вывод целых чисел без знака в двоичном представлении.....	331

10.9.2	Повышение переносимости и масштабируемости функции <code>displayBits</code>	333
10.9.3	Поразрядные операторы «И», «ИЛИ», исключающее «ИЛИ» и дополнение	334
10.9.4	Использование операторов поразрядного сдвига влево и вправо	337
10.9.5	Операторы поразрядного присваивания	339
10.10	Битовые поля	340
10.11	Константы-перечисления	343
10.12	Безопасное программирование на C	345
11	Файлы	347
11.1	Введение	348
11.2	Файлы и потоки данных	348
11.3	Создание файла с последовательным доступом	349
11.4	Чтение данных из файла с последовательным доступом	355
11.5	Произвольный доступ к файлам	360
11.6	Создание файла с произвольным доступом	361
11.7	Запись данных в файл с произвольным доступом	363
11.8	Чтение данных из файла с произвольным доступом	366
11.9	Пример: реализация программы для работы со счетами	368
11.10	Безопасное программирование на C	373
12	Структуры данных	375
12.1	Введение	376
12.2	Структуры, ссылающиеся на себя самих	377
12.3	Динамическое выделение памяти	377
12.4	Связанные списки	379
12.4.1	Функция <code>insert</code>	385
12.4.2	Функция <code>delete</code>	387
12.4.3	Функция <code>printList</code>	388
12.5	Стеки	388
12.5.1	Функция <code>push</code>	392
12.5.2	Функция <code>pop</code>	393
12.5.3	Области применения стеков	394
12.6	Очереди	395
12.6.1	Функция <code>enqueue</code>	399
12.6.2	Функция <code>dequeue</code>	400
12.7	Деревья	401
12.7.1	Функция <code>insertNode</code>	405

12 Содержание

12.7.2	Обход дерева: функции inOrder, preOrder и postOrder	406
12.7.3	Удаление дубликатов	407
12.7.4	Поиск в двоичных деревьях	407
12.8	Безопасное программирование на С	407
13	Препроцессор	409
13.1	Введение	410
13.2	Директива препроцессора #include	410
13.3	Директива препроцессора #define: символические константы	411
13.4	Директива препроцессора #define: макросы	412
13.5	Условная компиляция	414
13.6	Директивы препроцессора #error и #pragma	416
13.7	Операторы # и ##	416
13.8	Номера строк	417
13.9	Предопределенные символические константы	418
13.10	Утверждения	418
13.11	Безопасное программирование на С	419
14	Разное	420
14.1	Введение	421
14.2	Перенаправление ввода/вывода	421
14.3	Функции с переменным количеством аргументов	422
14.4	Использование аргументов командной строки	425
14.5	Замечания о компиляции программ из нескольких исходных файлов	426
14.6	Завершение выполнения программ с помощью функций exit и atexit	429
14.7	Окончания в литералах целых и вещественных чисел	430
14.8	Обработка сигналов	431
14.9	Динамическое выделение памяти: функции calloc и realloc	433
14.10	Безусловные переходы с помощью goto	434
A	Таблица предшествования операторов	437
B	Набор символов ASCII	439
C	Системы счисления	440
C.1	Введение	441
C.2	Преобразование двоичных чисел в восьмеричное и шестнадцатеричное представление	444

C.3	Преобразование восьмеричных и шестнадцатеричных чисел в двоичное представление	446
C.4	Преобразование двоичных, восьмеричных и шестнадцатеричных чисел в десятичное представление.....	446
C.5	Преобразование десятичных чисел в двоичное, восьмеричное и шестнадцатеричное представление.....	447
C.6	Отрицательные двоичные числа: нотация дополнения до двух	449
D	Сортировка: взгляд в глубину	451
D.1	Введение	452
D.2	Нотация «Большое O».....	452
D.3	Сортировка методом выбора.....	454
D.4	Сортировка методом вставки	457
D.5	Сортировка методом слияния.....	461
E	Дополнительные особенности стандарта C.....	468
E.1	Введение	469
E.2	Поддержка положений ревизии C99	470
E.3	Заголовочные файлы в C99	470
E.4	Включение объявлений в выполняемый код	471
E.5	Объявление переменных в заголовках инструкций for.....	472
E.6	Назначенные инициализаторы и составные литералы	473
E.7	Тип bool.....	476
E.8	Неявный тип int в объявлениях функций	477
E.9	Комплексные числа.....	479
E.10	Массивы переменной длины.....	480
E.11	Дополнительные возможности препроцессора	483
E.12	Другие особенности, определяемые ревизией C99	485
E.12.1	Минимальные требования компилятора к ресурсам	485
E.12.2	Ключевое слово restrict	485
E.12.3	Надежное целочисленное деление	486
E.12.4	Гибкие члены-массивы	486
E.12.5	Ослабление ограничений в составных инициализаторах	487
E.12.6	Математические операции обобщенного типа	487
E.12.7	Встраиваемые функции	488
E.12.8	Инструкция return без выражения.....	488
E.12.9	Предопределенный идентификатор __func__	488
E.12.10	Макрос va_copy	489
E.13	Новые особенности в ревизии C11	489

14 Содержание

E.13.1	Новые заголовочные файлы в C11.....	489
E.13.2	Поддержка многопоточной модели выполнения	490
E.13.3	Функция quick_exit.....	498
E.13.4	Поддержка Unicode®.....	498
E.13.5	Спецификатор функций _Noreturn.....	499
E.13.6	Выражения обобщенного типа	499
E.13.7	Аппекс L: анализируемость и неопределенное поведение....	499
E.13.8	Анонимные структуры и объединения	500
E.13.9	Управление выравниванием в памяти	501
E.13.10	Статические утверждения.....	501
E.13.11	Вещественные типы	501
E.14	Веб-ресурсы	501
F	Отладчик Visual Studio	505
F.1	Введение	506
F.2	Точки останова и команда Continue.....	506
F.3	Окна Locals и Watch	511
F.4	Управление выполнением с помощью команд Step Into, Step Over, Step Out и Continue.....	514
F.5	Окно Autos	517
G	Отладчик GNU.....	518
G.1	Введение	519
G.2	Точки останова и команды run, stop, continue и print.....	519
G.3	Команды print и set.....	525
G.4	Управление выполнением с помощью команд step, finish и next.....	527
G.5	Команда watch.....	530
	Алфавитный указатель.....	533

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.

1.1 Введение	1.6 Пробное приложение на языке C для Windows, Linux и Mac OS X
1.2 Язык программирования C	1.6.1 Запуск приложения из командной строки в Windows
1.3 Стандартная библиотека	1.6.2 Запуск приложения в Linux
1.4 C++ и другие C-подобные языки	1.6.3 Запуск приложения в Mac OS X
1.5 Типичная среда разработки приложений на языке C	1.7 Операционные системы
1.5.1 Фаза 1: создание программы	1.7.1 Windows – коммерческая операционная система
1.5.2 Фазы 2 и 3: препроцессинг и компиляция программы	1.7.2. Linux – открытая операционная система
1.5.3 Фаза 4: компоновка	1.7.3 Apple Mac OS X; Apple iOS® для устройств iPhone®, iPad® и iPod Touch®
1.5.4 Фаза 5: загрузка	1.7.4 Google Android
1.5.5 Фаза 6: выполнение	
1.5.6 Потоки стандартного ввода, стандартного вывода и стандартного вывода ошибок	

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 Деннисом 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 С 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 С развился в то, что теперь называют «традиционный С». 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 С, с высокими требованиями к производительности

Приложение	Описание
Операционные системы	Переносимость и производительность языка С 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 С. 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 С 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 С. – 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 национальном 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.

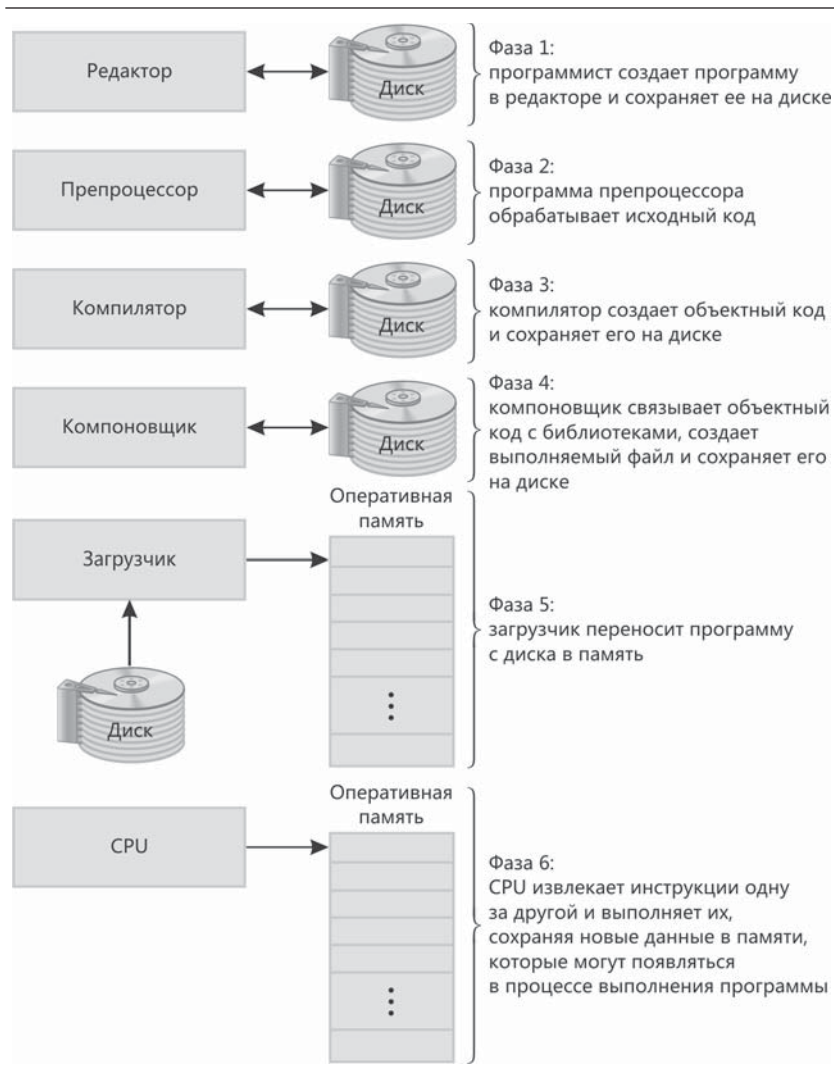


Рис. 1.1 | Типичная среда разработки приложений на языке C

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 compiler¹). Чтобы скомпилировать

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

и скомпилировать 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 скомпилирована 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 Поток 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.

Компилировать 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 рубленным 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.

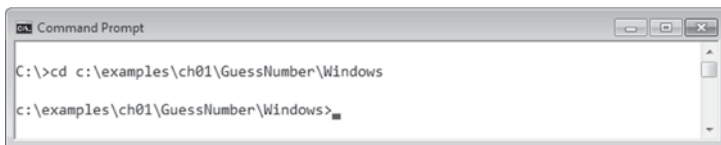


Рис. 1.2 | Откройте окно **Command Prompt (Командная строка)** и перейдите в каталог приложения

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 попытается добавить расширение `.exe`, если оно не указано.]

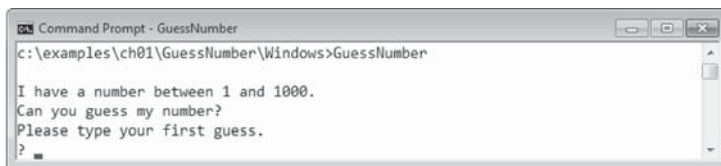


Рис. 1.3 | Запустите приложение *GuessNumber*

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).

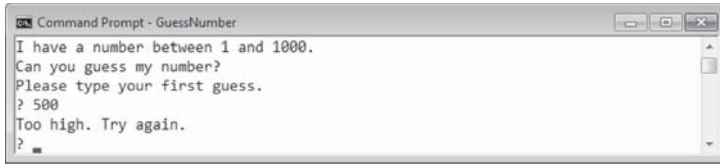


Рис. 1.4 | Введите первое число

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.

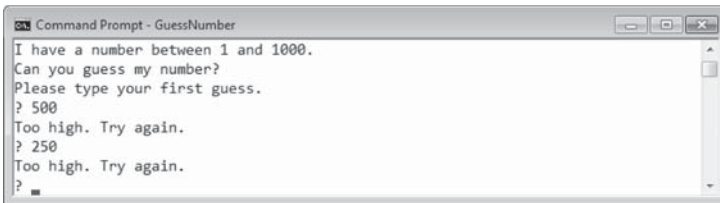


Рис. 1.5 | Введите второе число и получите ответ

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.

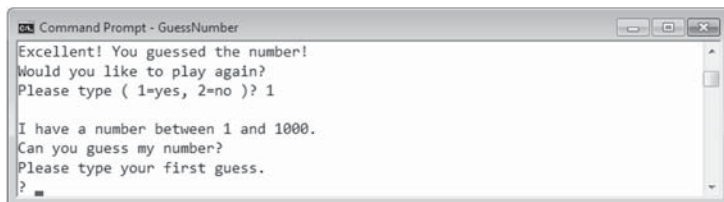
8. Закройте окно *Command Prompt* (Командная строка).



```
Command Prompt - GuessNumber
Too high. Try again.
? 125
Too high. Try again.
? 62
Too high. Try again.
? 31
Too low. Try again.
? 46
Too high. Try again.
? 39
Too low. Try again.
? 43
Too high. Try again.
? 41
Too low. Try again.
? 42

Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? █
```

Рис. 1.6 | Продолжайте попытки, пока не угадаете число



```
Command Prompt - GuessNumber
Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 1

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? █
```

Рис. 1.7 | Сыграйте еще раз



```
Command Prompt
Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 2

C:\examples\ch01\GuessNumber\Windows> █
```

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).