Введение

C является языком программирования общего назначения. Он всегда находился в самой тесной связи с операционной системой Unix, в которой и для которой он и разрабатывался, поскольку как сама система, так и большинство работающих в ней программ написаны именно на C. Тем не менее сам язык не привязан жестко к одной определенной системе или аппаратной платформе. И хотя C называют "языком системного программирования", поскольку на нем удобно писать компиляторы и операционные системы, он столь же удобен и для написания больших прикладных программ в самых разных областях применения.

Многие ключевые идеи C пришли из языка BCPL, разработанного Мартином Ричардсом (Martin Richards). BCPL оказал влияние на C опосредованно — через язык B, разработанный Кеном Томпсоном (Ken Thompson) в 1970 году для первой системы Unix на ЭВМ DEC PDP-7.

В языках BCPL и B отсутствует типизация данных. В отличие от них язык C предлагает широкий ассортимент типов. Фундаментальные типы включают в себя символьный, а также целый и вещественный (с плавающей точкой) нескольких различных размеров. Кроме того, существует целая иерархия производных типов данных, создаваемых с помощью указателей, массивов, структур и объединений. Из операндов и знаков операций формируются выражения; любое выражение, в том числе присваивание и вызов функции, может являться оператором. Благодаря указателям поддерживается механизм системно-независимой адресной арифметики.

В языке C имеются все основные управляющие конструкции, необходимые для написания хорошо структурированной программы: группировка операторов в блоки, принятие решения по условию (`if-else`), выбор одного из нескольких возможных вариантов (`switch`), циклы с проверкой условия завершения в начале (`while`, `for`) и в конце (`do`), а также принудительный выход из цикла (`break`).

Функции могут возвращать значения простых типов, структуры, объединения или указатели. Любую функцию можно вызывать рекурсивно. Локальные переменные функции обычно являются "автоматическими", т.е. создаются при каждом ее вызове. Определения (тела) функций нельзя вкладывать друг в друга, однако переменные можно объявлять в блочно-структурированном стиле. Функции программы на C могут находиться в отдельных файлах исходного кода, компилируемых также отдельно. Переменные могут быть внутренними по отношению к функции, внешними и при этом видимыми только в одном файле кода или же видимыми во всем пространстве программы.

На этапе препроцессорной обработки в тексте программы выполняются макроподстановки, включение дополнительных файлов исходного кода и условная компиляция.

C — это язык сравнительно "низкого" уровня. Ничего уничижительного в этом определении нет; это всего лишь значит, что язык C работает с теми же объектами, что и большинство компьютерных систем, а именно с символами, числами и адресами. Эти данные можно комбинировать разными способами с помощью арифметических и логических операций, которые реализованы реальными аппаратными и системными средствами.

В языке C нет операций для прямого манипулирования составными объектами, например строками символов, множествами, списками или массивами. Не существует операций для непосредственной работы с целым массивом строк, хотя структуру можно

скопировать как единое целое. Язык не содержит специальных средств распределения памяти — только статическое определение и стек, в котором хранятся локальные переменные функций; не предусмотрена ни системная куча (*heap*), ни сборка мусора (*garbage collection*). Наконец, в самом языке нет и средств ввода-вывода наподобие операторов `READ` или `WRITE`, а также отсутствуют встроенные механизмы обращения к файлам. Все эти операции высокого системного уровня выполняются путем явного вызова функций. В большинстве реализаций языка C имеется более или менее стандартный набор таких функций.

Аналогично, в C имеются только несложные, однопоточные управляющие структуры: проверки условий, циклы, блоки и подпрограммы, однако отсутствует мультипрограммирование, распараллеливание операций, синхронизация и сопрограммы.

Хотя отсутствие некоторых из перечисленных средств может показаться очень серьезным недостатком (“То есть я должен вызывать функцию, чтобы просто сравнить две строки символов?!”), тем не менее в том, чтобы сохранять набор базовых конструкций языка относительно небольшим, есть и свои преимущества. Поскольку C невелик, его описание занимает немного места, а изучение отнимает немного времени. Каждый программист вполне может знать, понимать и регулярно использовать практически всю базу языка.

Много лет определением языка служил справочник по нему, включенный в первое издание книги *Язык программирования C*. В 1983 году Американский национальный институт стандартов (ANSI) основал комитет для создания полного и современного определения языка C. В результате к концу 1988 года было завершено создание стандарта ANSI C. Большинство средств и возможностей этого стандарта поддерживается современными компиляторами.

Стандарт основан на первоначальном справочнике по языку. Сам язык изменился относительно мало; одной из целей его стандартизации было обеспечить совместимость с большинством уже написанных программ или по крайней мере добиться от компиляторов корректных предупреждений, если там будут встречаться новые средства.

Для большинства программистов самым важным новшеством оказался новый синтаксис объявления и определения функций. Объявление функции теперь может содержать описание ее аргументов; синтаксис определения должен соответствовать объявлению. Дополнительная информация сильно облегчает компиляторам работу по выявлению ошибок, причиной которых стала несогласованность типов аргументов. Судя по нашему опыту, это очень полезное дополнение к языку.

Есть и другие небольшие изменения. Присваивание структур и перечислимые типы, которые давно уже встречались в разных реализациях, официально стали частью языка. Операции с плавающей точкой теперь можно выполнять с одинарной точностью. Более четко определены свойства арифметических операций, особенно над переменными без знака. Препроцессор стал более сложным и развитым. Однако б*о*льшая часть изменений не очень повлияет на работу программистов.

Второе существенное новшество, связанное с введением стандарта, — это определение стандартной библиотеки функций, включаемой в состав компилятора C. В эту библиотеку входят функции для обращения к операционной системе (например, для чтения и записи файлов), для форматированного ввода-вывода, распределения памяти, операций со строками и т.п. Совокупность стандартных заголовочных файлов обеспечивает единообразие обращения к объявлениям функций и типов данных. Программы, которые пользуются этой библиотекой для взаимодействия с операционной системой, являются

гарантированно совместимыми с разными системными средами. Бо́льшая часть библиотеки построена по принципам стандартной библиотеки ввода-вывода системы Unix. Эта библиотека описывалась еще в первом издании и с тех пор широко использовалась во многих других системах. Большинство программистов и здесь не столкнутся с существенными изменениями.

Поскольку типы данных и управляющие структуры языка С поддерживаются многими компьютерами непосредственно на аппаратном уровне, библиотека функций, необходимых для реализации автономных, полнофункциональных программ, имеет совсем крошечные размеры. Функции из стандартной библиотеки всегда вызываются только явным образом, так что их применения легко избежать, если это необходимо. Большинство функций написаны на С и полностью переносимы, за исключением некоторых подробностей работы операционных систем, которые скрыты на более низком уровне.

Хотя язык С соответствует по своей структуре аппаратным возможностям многих компьютеров, он не привязан к какой-либо конкретной машинной архитектуре. При некотором усердии на нем легко написать полностью переносимую программу, т.е. программу, которая будет работать безо всяких изменений на самых разных компьютерах. В стандарте вопросы переносимости четко определены и выведены на явный уровень. Стандартом предписывается использование набора констант, характеризующего архитектуру системы, в которой работает программа.

Языку С не свойствен строгий контроль типов, хотя по мере его эволюции развивались средства проверки соответствия типов. В исходной версии С компилятор выдавал предупреждения, но не запрещал взаимную замену указателей и целых переменных разных типов. Но от этого уже давно отказались, и теперь стандарт требует полной совместимости объявлений и явного выполнения преобразований типов. Хорошие компиляторы уже давно требовали от программиста того же самого. Еще одним шагом в этом направлении как раз и является новый стиль объявления функций. Компиляторы выдают предупреждения по поводу большинства ошибок, связанных с несоответствием типов, и не выполняется никакого автоматического преобразования несовместимых типов данных. И все же в С сохраняется традиционное отношение к программисту, как к человеку, который сам должен знать и решать, что ему делать; от него просто требуется выражать свои намерения в явной и четкой форме.

Как и у любых языков, у С есть свои недостатки: некоторые операции имеют нелогичный приоритет; некоторые синтаксические конструкции можно было бы организовать и получше. И тем не менее язык С доказал свою высочайшую эффективность и выразительность для самого широкого круга приложений.

Эта книга организована следующим образом. Глава 1 представляет собой краткое учебное пособие по основным конструкциям С. Цель этой главы — быстро ввести читателя в курс дела. Мы убеждены, что лучший способ изучить новый язык — это сразу начать писать программы на нем. В этом учебном пособии предполагается, что читатель имеет некоторые знания по основам программирования, поэтому мы не разжевываем, что такое компьютер, зачем нужен компилятор или что означает выражение `n=n+1`. Хотя мы и старались везде, где это возможно, демонстрировать полезные приемы программирования, все-таки книга не является пособием по структурам данных и алгоритмам. Если необходимо было сделать подобный выбор, мы всегда сосредоточивались на вопросах, связанных с самим языком.

В главах 2–6 более подробно и строго, чем в главе 1, рассматриваются различные аспекты языка С, хотя главное место в изложении по-прежнему занимают примеры закон-

ченных программ, а не изолированные фрагменты кода. В главе 2 рассматриваются базовые типы данных, операции и выражения. В главе 3 обсуждаются управляющие конструкции: `if-else`, `switch`, `while`, `for` и т.д. Глава 4 посвящена функциям и структуре программы: внешним переменным, области видимости, распределению кода по исходным файлам и т.п. Также в ней рассматривается работа с препроцессором. В главе 5 описаны указатели и операции адресной арифметики. В главе 6 рассматриваются структуры и объединения.

Глава 7 содержит описание стандартной библиотеки функций, представляющей собой стандартный интерфейс для взаимодействия с операционной системой. Эта библиотека определена стандартом ANSI и должна поддерживаться во всех системах, в которых реализован язык C, так что программы, которые пользуются ее функциями для ввода, вывода и других операций на уровне операционной системы, должны обладать свойством переносимости на другие платформы безо всяких изменений.

В главе 8 описывается взаимодействие программ на C и операционной системы Unix. Основное внимание уделяется вводу-выводу, файловой системе и распределению памяти. Хотя частично эта глава содержит только информацию, специфическую для систем Unix, программисты, работающие с другими системами, все равно найдут здесь для себя много полезного. Среди прочего это взгляд изнутри на реализацию одной из версий стандартной библиотеки, а также рекомендации по переносимости программ.

Приложение A является полным справочником по языку. Официальным документом по синтаксису и семантике языка C является сам стандарт ANSI. Однако стандарт как справочное пособие в основном предназначается для разработчиков компиляторов, в то время как справочник в этой книге описывает язык более доходчиво, кратко и без перегруженности официальными выражениями. В приложении Б содержится описание стандартной библиотеки, опять-таки предназначенное скорее для программистов-пользователей, чем для разработчиков компиляторов. В приложение В включена сводка изменений, внесенных в язык по сравнению с его исходной версией. Однако в случае возникновения каких-либо сомнений и разночтений программисту всегда следует опираться на стандарт и имеющийся в наличии компилятор.