# А. Справочное руководство

## А 1. Введение

Данное руководство описывает язык программирования Си, определенный 31 октября 1989 г. в соответствии с проектом, утвержденным в ANSI в качестве Американского национального стандарта для информационных систем: Язык программирования Си, Х3.159-1989 ("American National Standard for Information Systems — Programming Language C, X3.159-1989"). Это описание — лишь один из вариантов предлагаемого стандарта, а не сам стандарт, однако мы специально заботились о том, чтобы сделать его надежным руководством по языку.

Настоящий документ в основном следует общей схеме описания, принятой в стандарте (публикация которого в свою очередь основывалась на первом издании этой книги), однако в организационном плане есть различия. Если не считать отклонений в названиях нескольких продуктов и отсутствия формальных определений лексем и препроцессора, грамматика языка здесь и грамматика в стандарте эквивалентны.

Далее примечания (как и это) набираются с отступом от левого края страницы. В основном эти примечания касаются отличий стандарта от версии языка, описанной в первом издании этой книги, и от последующих нововведений в различных компиляторах.

## А 2. Соглашения о лексике

Программа состоит из одной или нескольких *единиц трансляции*, хранящихся в виде файлов. Каждая такая единица проходит несколько фаз трансляции, описанных в А12. Начальные фазы осуществляют лексические преобразования нижнего уровня, выполняют директивы, заданные в программе строками, начинающимися со знака #, обрабатывают макроопределения и производят макрорасширения. По завершении работы препроцессора (А12) программа представляется в виде последовательности лексем.

### А 2.1. Лексемы (tokens)

Существуют шесть классов лексем (или токенов): идентификаторы, ключевые слова, константы, строковые литералы, операторы и прочие разделители. Пробелы, горизонтальные и вертикальные табуляции, новые строки, переводы страницы и комментарии (имеющие общее название символы-разделители) рассматриваются компилятором только как разделители лексем и в остальном на результат трансляции влияния не оказывают. Любой из символов-разделителей годится, чтобы отделить друг от друга соседние идентификаторы, ключевые слова и константы.

Если входной поток уже до некоторого символа разбит на лексемы, то следующей лексемой будет самая длинная строка, которая может быть лексемой.

### А 2.2. Комментарий

Символы /\* открывают комментарий, а символы \*/ закрывают его. Комментарии нельзя вкладывать друг в друга, их нельзя помещать внутрь строк или текстовых литералов.

### А 2.3. Идентификаторы

Идентификатор — это последовательность букв и цифр. Первым символом должна быть буква; знак подчеркивания \_ считается буквой. Буквы нижнего и верхнего регистров различаются. Идентификаторы могут иметь любую длину; для внутренних идентификаторов значимыми являются первые 31 символ; в некоторых реализациях принято большее число значимых символов. К внутренним идентификаторам относятся имена макросов и все другие имена, не имеющие внешних связей (А11.2). На идентификаторы с внешними связями могут накладываться большие ограничения: иногда воспринимаются не более шести первых символов и могут не различаться буквы верхнего и нижнего регистров.

### А 2.4. Ключевые слова

Следующие идентификаторы зарезервированы в качестве ключевых слов и в другом смысле использоваться не могут:

|  |  |  |  |
| --- | --- | --- | --- |
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| const | float | short | unsigned |
| continue | for | signed | void |
| default | goto | sizeof | volatile |
| do | if | static | while |

В некоторых реализациях резервируются также слова fortran и asm.

Ключевые слова const, signed и volatile впервые появились в стандарте ANSI; enum и void — новые по отношению к первому изданию книги, но уже использовались; ранее зарезервированное entry нигде не использовалось и поэтому более не резервируется.

### А 2.5. Константы

Существует несколько видов констант. Каждая имеет свой тип данных; базовые типы рассматриваются в А4.2.

*константа:*

*целая-канстанта символьная-константа*

*константа-с-плавающей-точкой константа-перечисление*

#### А 2.5.1. Целые константы

Целая константа, состоящая из последовательности цифр, воспринимается как восьмеричная, если она начинается с 0 (цифры нуль), и как десятичная в противном случае. Восьмеричная константа не содержит цифр 8 и 9. Последовательность цифр, перед которой стоят 0х или 0Х, рассматривается как шестнадцатеричное целое. В шестнадцатеричные цифры включены буквы от а (или А) до f (или F) со значениями от 10 до 15.

Целая константа может быть записана с буквой-суффиксом u (или U) для спецификации ее как беззнаковой константы. Она также может быть с буквой-суффиксом l (или L) для указания, что она имеет тип long.

Тип целой константы зависит от ее вида, значения и суффикса (о типах см. А4). Если константа — десятичная и не имеет суффикса, то она принимает первый из следующих типов, который годится для представления ее значения: int, long int, unsigned long int. Восьмеричная или шестнадцатеричная константа без суффикса принимает первый возможный из типов: int, unsigned int, long int, unsigned long int. Если константа имеет суффикс u или U, то она принимает первый возможный из типов: unsigned int, unsigned long int. Если константа имеет суффикс l или L,то она принимает первый возможный из типов: long int, unsigned long int. Если константа имеет суффикс ul или UL, то она принимает тип unsigned long int.

Типы целых констант получили существенное развитие в сравнении с первой редакцией языка, в которой большие целые имели просто тип long. Суффиксы U и u введены впервые.

#### А 2.5.2. Символьные константы

Символьная константа — это последовательность из одной или нескольких символов, заключенная в одиночные кавычки (например 'х'). Если внутри одиночных кавычек расположен один символ, значением константы является числовое значение этого символа в кодировке, принятой на данной машине. Значение константы с несколькими символами зависит от реализации.

Символьная константа не может содержать в себе одиночную кавычку ' или символ новой строки; чтобы изобразить их и некоторые другие символы, могут быть использованы эскейп-последовательности:

|  |  |  |
| --- | --- | --- |
| новая строка (newline, linefeed) | NL (LF) | \n |
| горизонтальная табуляция (horisontal tab) | HT | \t |
| вертикальная табуляция (vertical tab) | VT | \v |
| возврат на шаг (backspace) | BS | \b |
| возврат каретки (carriage return) | CR | \r |
| перевод страницы (formfeed) | FF | \f |
| сигнал звонок (audible alert, bell) | BEL | \a |
| обратная наклонная черта (backslash) | \ | \\ |
| знак вопроса (question mark) | ? | \? |
| одиночная кавычка (single quote) |  | \' |
| двойная кавычка (double quote) |  | \" |
| восьмеричный код (octal number) | *ooo* | \*ooo* |
| шестнадцатеричный код (hex number) | *hh* | \x*hh* |

Эскейп-последовательность \*ооо* состоит из обратной наклонной черты, за которой следуют одна, две или три восьмеричные цифры, специфицирующие значение желаемого символа. Наиболее частым примером такой конструкции является \0 (за которой не следует цифра); она специфицирует NULL-символ. Эскейп- последовательность \x*hh* состоит из обратной наклонной черты с буквой х, за которыми следуют шестнадцатеричные цифры, специфицирующие значение желаемого символа. На количество цифр нет ограничений, но результат будет не определен, если значение полученного символа превысит значение самого "большого" из допустимых символов. Если в данной реализации тип char трактуется как число со знаком, то значение и в восьмеричной, и в шестнадцатеричнои эскейп-последовательности получается с помощью "распространения знака", как если бы выполнялась операция приведения к типу char. Если за \ не следует ни один из перечисленных выше символов, результат не определен.

В некоторых реализациях имеется расширенный набор символов, который не может быть охвачен типом char. Константа для такого набора пишется с буквой L впереди (например, L'х') и называется расширенной символьной константой. Такая константа имеет тип wchar\_t (целочисленный тип, определенный в стандартном заголовочном файле <stddef.h>). Как и в случае обычных символьных констант, здесь также возможны восьмеричные и шестнадцатеричные эскейп-последовательности; если специфицированное значение превысит тип wchar\_t, результат будет не определен.

Некоторые из приведенных эскейп-последовательностей новые (шестнадцатеричные в частности). Новым является и расширенный тип для символов. Наборам символов, обычно используемым в

Америке и Западной Европе, подходит тип char, а тип wchar\_t был добавлен главным образом для азиатских языков.

#### А 2.5.3. Константы с плавающей точкой

Константа с плавающей точкой состоит из целой части, десятичной точки, дробной части, е или Е и целого (возможно, со знаком), представляющего порядок, и, возможно, суффикса типа, задаваемого одной из букв: f, F, l или L. И целая, и дробная часть представляют собой последовательность цифр. Либо целая часть, либо дробная часть (но не обе вместе) могут отсутствовать; также могут отсутствовать десятичная точка или Е с порядком (но не обе одновременно). Тип определяется суффиксом; F или f определяют тип float, L или l

— тип long double; при отсутствии суффикса подразумевается тип double. Суффиксы для констант с плавающей точкой являются нововведением.

#### А 2.5.4. Константы-перечисления

Идентификаторы, объявленные как элементы перечисления (А8.4), являются константами типа int.

### А 2.6. Строковые литералы

Строковый литерал, который также называют строковой константой, — это последовательность символов, заключенная в двойные кавычки (например, "..."). Строка имеет тип "массив символов" и память класса static (А4), которая инициализируется заданными символами. Представляются ли одинаковые строковые литералы одной копией или несколькими, зависит от реализации. Поведение программы, пытающейся изменить строковый литерал, не определено.

Написанные рядом строковые литералы объединяются (конкатенируются) в одну строку. После любой конкатенации к строке добавляется NULL-байт (\0), что позволяет программе, просматривающей строку, найти ее конец. Строковые литералы не могут содержать в себе символ новой строки или двойную кавычку; в них нужно использовать те же эскейп-последовательности, что и в символьных константах.

Как и в случае с символьными константами, строковый литерал с символами из расширенного набора должен начинаться с буквы L (например L"..."). Строковый литерал из расширенного набора имеет тип "массив из wchar\_t". Конкатенация друг с другом обычных и "расширенных" строковых литералов не определена.

То, что строковые литералы не обязательно представляются разными копиями, запрет на их модификацию, а также конкатенация соседних строковых литералов — нововведения ANSI-стандарта. "Расширенные" строковые литералы также объявлены впервые.