**Функция scanf**

#include <stdio.h>

int scanf(const char \*format, ...);

Функция scanf() представляет собой процедуру ввода общего назначения, которая читает поток stdin и сохраняет информацию в переменных, перечисленных в списке аргументов. Она может читать все встроенные типы данных и автоматически преобразовывать их в соответствующий внутренний формат.

В версии C99 к параметру format применен квалификатор restrict.

Управляющая строка, задаваемая параметром format, состоит из символов трех категорий:

спецификаторов формата;

пробельных символов;

символов, отличных от пробельных.

Спецификации формата начинаются знаком % и сообщают функции scanf() тип данного, которое будет прочитано. Спецификации формата приведены в таблице 13.3. Например, по спецификации %s будет прочитана строка, а по спецификации %d — целое значение. Строка форматирования читается слева направо, и спецификации формата сопоставляются аргументам в порядке их перечисления в списке аргументов.

Таблица 13.3. Спецификации формата функции scanf() Код Назначение

%a Читает значение с плавающей точкой (только C99)

%A Аналогично коду %a (только C99)

%c Читает один символ

%d Читает десятичное целое

%i Читает целое в любом формате (десятичное, восьмеричное или шестнадцатеричное)

%e Читает число с плавающей точкой

%E Аналогично коду %e

%f Читает число с плавающей точкой

%F Аналогично коду %f (только С99)

%g Читает число с плавающей точкой

%G Аналогично коду %g

%o Читает восьмеричное число

%s Читает строку

%x Читает шестнадцатеричное число

%X Аналогично коду %x

%p Читает указатель

%n Принимает целое значение, равное количеству прочитанных до сих пор символов

%u Читает десятичное целое без знака

%[] Просматривает набор символов

%% Читает знак процента

По умолчанию спецификации a, f, e и g заставляют функцию scanf() присваивать данные переменным типа float. Если перед одной из этих спецификаций поставить модификатор l, функция scanf() присвоит прочитанные данные переменной типа double. Использование же модификатора L означает, что полученное значение присвоится переменной типа long double.

Современные компиляторы, поддерживающие добавленные в 1995 году средства работы с двухбайтовыми символами, позволяют к спецификации c применить модификатор l; тогда будет считаться, что соответствующий указатель указывает на двухбайтовый символ (т.е. на данное типа whcar\_t). Модификатор l также можно использовать с кодом формата s; тогда будет считаться, что соответствующий указатель указывает на строку двухбайтовых символов. Кроме того, модификатор l можно использовать для того, чтобы указать, что набор сканируемых символов состоит из двухбайтовых символов.

Если в строке форматирования встретится разделитель, то функция scanf() пропустит один или несколько разделителей во входном потоке. Под разделителем, или пробельным символом, подразумевается пробел, символ табуляции или разделитель строк (символ новой строки). По сути, наличие одного разделителя в управляющей строке приведет к тому, что функция scanf() будет читать, не сохраняя, любое количество (возможно, даже нулевое) разделителей до первого символа, отличного от разделителя.

Если в строке форматирования встретился символ, отличный от разделителя, то функция scanf() прочитает и отбросит его. Например, если в строке форматирования встретится %d, %d, то функция scanf() сначала прочитает целое значение, затем прочитает и отбросит запятую и, наконец, прочитает еще одно целое. Если заданный символ не найден, функция scanf() завершает работу.

Все переменные, получающие значения с помощью функции scanf(), должны передаваться посредством своих адресов. Это значит, что все аргументы должны быть указателями на переменные.

Элементы входного потока должны быть разделены пробелами, символами табуляции или разделителями строк. Такие символы, как запятая, точка с запятой и т.п., не распознаются в качестве разделителей. Это означает, что оператор

scanf("%d%d", &r, &c);

примет значения, введенные как 10 20, но откажется от последовательности символов 10, 20.

Символ \*, стоящий после знака % и перед кодом формата, прочитает данные заданного типа, но запретит их присваивание. Следовательно, оператор

scanf("%d%\*c%d", &x, &y);

при вводе данных в виде 10/20 поместит значение 10 в переменную x, отбросит знак деления и присвоит значение 20 переменной у.

Команды форматирования могут содержать модификатор максимальной длины поля. Он представляет собой целое число, располагаемое между знаком % и кодом формата, которое ограничивает количество читаемых для всех полей символов. Например, если в переменную address нужно прочитать не более 20 символов, используется следующий оператор.

scanf("%20s", address);

Если входной поток содержит более 20 символов, то при последующем обращении к операции ввода чтение начнется с того места, в котором "остановился" предыдущий вызов функции scanf(). Если разделитель встретится раньше, чем достигнута максимальная длина поля, ввод данных завершится. В этом случае функция scanf() переходит к чтению следующего поля.

Хотя пробелы, символы табуляции и разделители строк используются в качестве разделителей полей, при чтении одиночного символа они читаются подобно любому другому символу. Например, если входной поток состоит из символов x у, то оператор

scanf("%c%c%c", &a, &b, &c);

поместит символ x в переменную а, пробел — в переменную b, а символ у — в переменную с.

Помните, что любые символы управляющей строки (включая пробелы, символы табуляции и новой строки), не являющиеся спецификациями формата, используются для установки соответствия и отбрасывания символов из входного потока. Любой соответствующий им символ отбрасывается. Например, если поток ввода выглядит, как 10t20, оператор

scanf("%dt%d", &x, &y);

присвоит переменной x значение 10, а переменной у — значение 20. Символ t отбрасывается, так как он присутствует в управляющей строке.

Функция scanf() поддерживает спецификатор формата общего назначения, называемый набором сканируемых символов (scanset). В этом случае определяется набор символов, которые могут быть прочитаны функцией scanf() и присвоены соответствующему массиву символов. Для определения такого набора символы, подлежащие сканированию, необходимо заключить в квадратные скобки. Открывающая квадратная скобка должна следовать сразу за знаком процента. Например, следующий набор сканируемых символов указывает на то, что необходимо читать только символы A, B и C.

%[ABC]

При использовании набора сканируемых символов функция scanf() продолжает читать символы и помещать их в соответствующий массив символов до тех пор, пока не встретится символ, отсутствующий в заданном наборе. Соответствующая набору переменная должна быть указателем на массив символов. При возврате из функции scanf() этот массив будет содержать строку из прочитанных символов, завершающуюся символом конца строки.

Если первый символ в наборе является знаком ^, то получаем обратный эффект: входное поле читается до тех пор, пока не встретится символ из заданного набора сканируемых символов, т.е. знак ^ заставляет функцию scanf() читать только те символы, которые отсутствуют в наборе сканируемых символов.

Во многих реализациях допускается задавать диапазон с помощью дефиса. Например, функция scanf(), встречая набор сканируемых символов в виде %[A-z], будет читать символы, попадающие в диапазон от А до Z.

Важно помнить, что в наборе сканируемых символов различаются прописные и строчные буквы. Следовательно, чтобы сканировать как прописные, так и строчные буквы, в наборе сканируемых символов придется задать их отдельно.

Функция scanf() возвращает число, равное количеству полей, для которых успешно присвоены значения. К этим полям не относятся поля, которые были прочитаны, но присвоение не состоялось в связи с использованием модификатора \*, подавляющего присваивание. При обнаружении ошибки до присвоения значения первого поля функция scanf() возвращает значение EOF.

Модификаторы формата, добавленные к функции scanf() Стандартом C99

В версии C99 для использования в функции scanf() добавлены модификаторы формата hh, ll, j, z и t. Модификатор hh можно применять к спецификациям d, i, о, u, x и n. Он означает, что соответствующий аргумент является указателем на значение типа signed char или unsigned char. Модификатор ll также можно применять к спецификациям d, i, о, u, x и n. Он означает, что соответствующий аргумент является указателем на значение типа signed long long int или unsigned long long int.

Moдификaтop фopмaтa j, кoтopый пpимeняeтcя к cпeцификaциям d, i, о, u, х и n означает, что соответствующий аргумент является указателем на значение типа intmax\_t или uintmax\_t. Эти типы объявлены в заголовке <stdint.h> и служат для хранения целых максимально возможной разрядности.

Модификатор формата z, который применяется к спецификациям d, i, о, u, х и n означает, что соответствующий аргумент является указателем на объект типа size\_t. Этот тип объявлен в заголовке <stddef.h> и служит для хранения результата операции sizeof.

Модификатор формата t, который применяется к спецификациям d, i, о, u, х и n означает, что соответствующий аргумент является указателем на объект типа ptrdiff\_t. Этот тип объявлен в заголовке <stddef.h> и служит для хранения значения разности между двумя указателями.

Пример

Действие данных операторов scanf() объясняется в комментариях.

#include <stdio.h>

int main(void)

{

char str[80], str2[80];

int i;

/\* читается строка и целое значение \*/

scanf("%s%d", str, &i);

/\* в переменную str считывается не более 79 символов \*/

scanf("%79s", str);

/\* целое, расположенное между двумя строками, пропускается \*/

scanf("%s%\*d%s", str, str2);

return 0;

}