Naming constants in a program is called as **macro definition**, #define CONSTANT\_NAME “value”

If the value is like this, (5.0f/9.0f) then do not avoid append f because then C may round off the value of 5/9.

Identifiers are names of variables, functions, macros, etc. The identifiers can contain only alphabets, numbers and underscore. It cannot begin with number or underscore.

C is case sensitive therefore dog, Dog, DoG are all different variables.

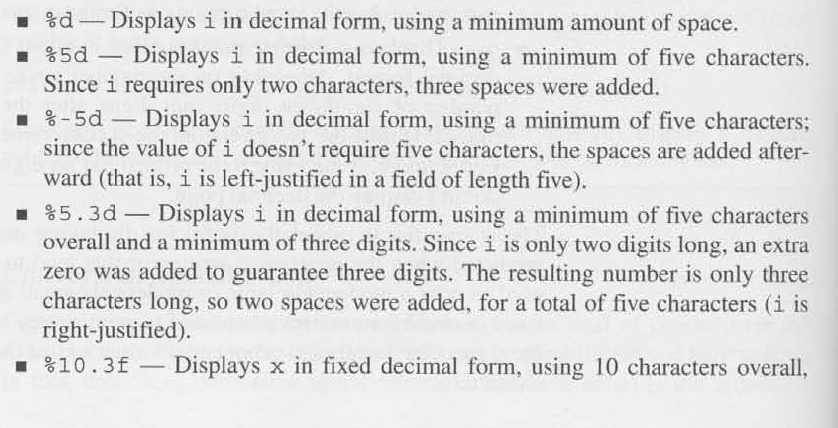
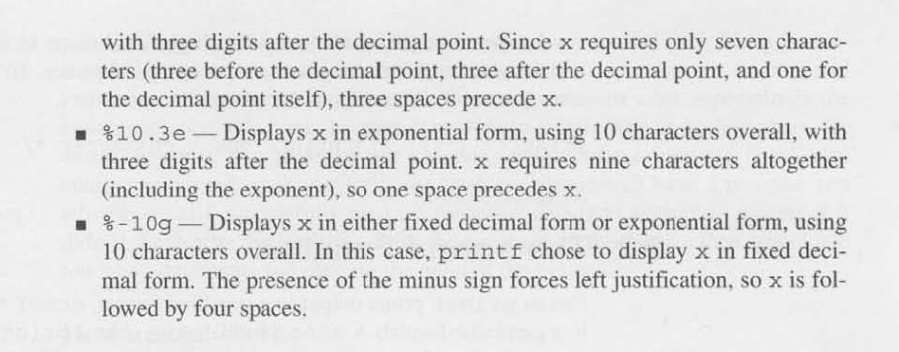
Keywords are words which hold a special meaning in the program and cannot be used as an identifier.

int a; -> this is **declaration**

int a = 10; -> this is **initialization** as we assign a value as the variable is declared

%d, %f, %g, %e is called as conversion specifiers that are to be used in printf to display values, as they convert the values from binary to the required format mentioned by the letter.

A conversion specification is of the form %m.pX where m is the **minimum field width** and p is the **precision** which depends on the **conversion specifier** i.e., X.

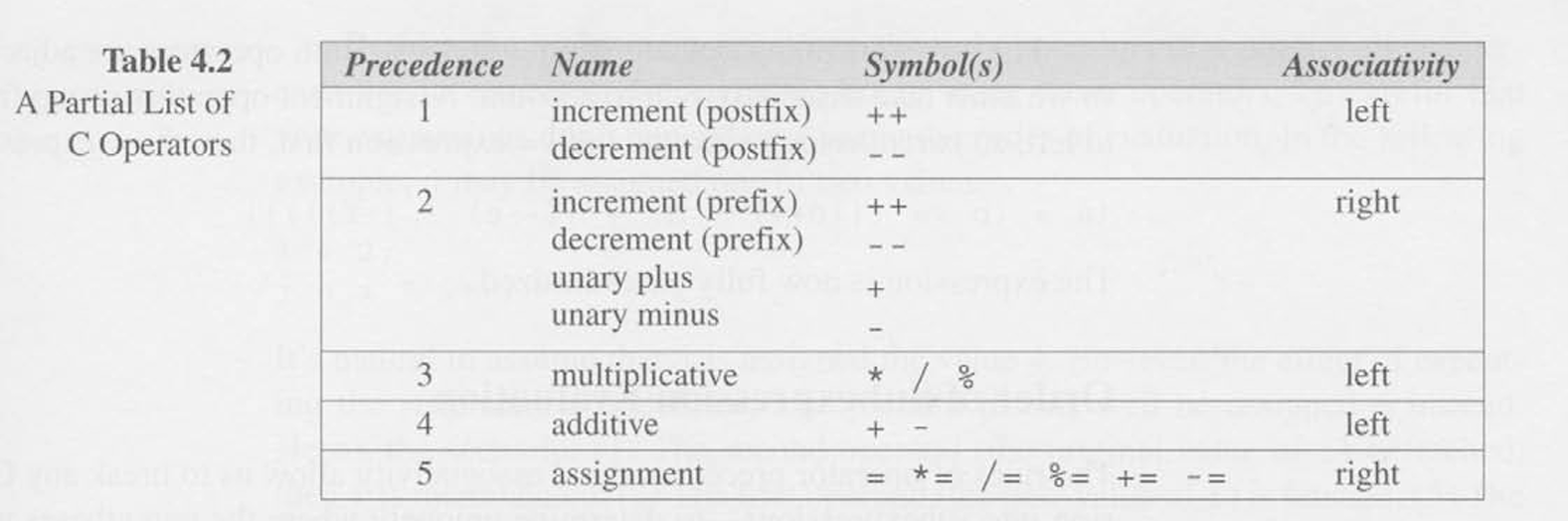
For example, if there is a number 123 then using %4d would right justify it like .123 (the full stop represents the space) and %-4d would left justify it like 123.

Just like Python, the C language has escape sequences which are used to represent characters which otherwise have special meaning in the language.

In C,

1. \a – audible beep
2. \t - tab
3. \b – moves cursor back one position
4. \n – new line

We don’t use \n at the end of scanf.

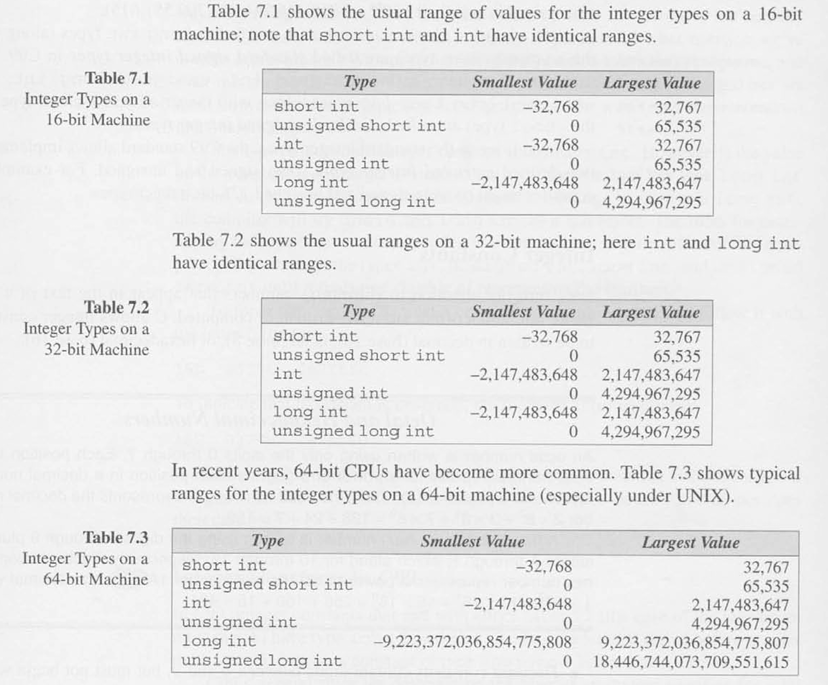


The precedence of relational operators (>, <, >=, <=) is lower than the arithmetic operators and they are also left associative.

The precedence of equality operators (==,!=) is lower than relational operators and they are left associative.

For logical operators,  
! represents logical not  
&& represents logical and  
|| represents logical or  
the condition to the left of the logical operators is always evaluated first.  
‘and’ and ‘not’ have lower precedence than equality and is left associative while ‘!’ has same precedence as unary arithmetic operators and is right associative

**Dangling else:** when nested if statements are used and else statement is used without proper braces then the else statement is associated with the nearest if statement and doesn’t depend on the indentation.

  
the ranges defined above are not according to the C standards and depends on the compiler being used, one way to check the ranges is by using the *<limit.h>* header and the macros defined by it.

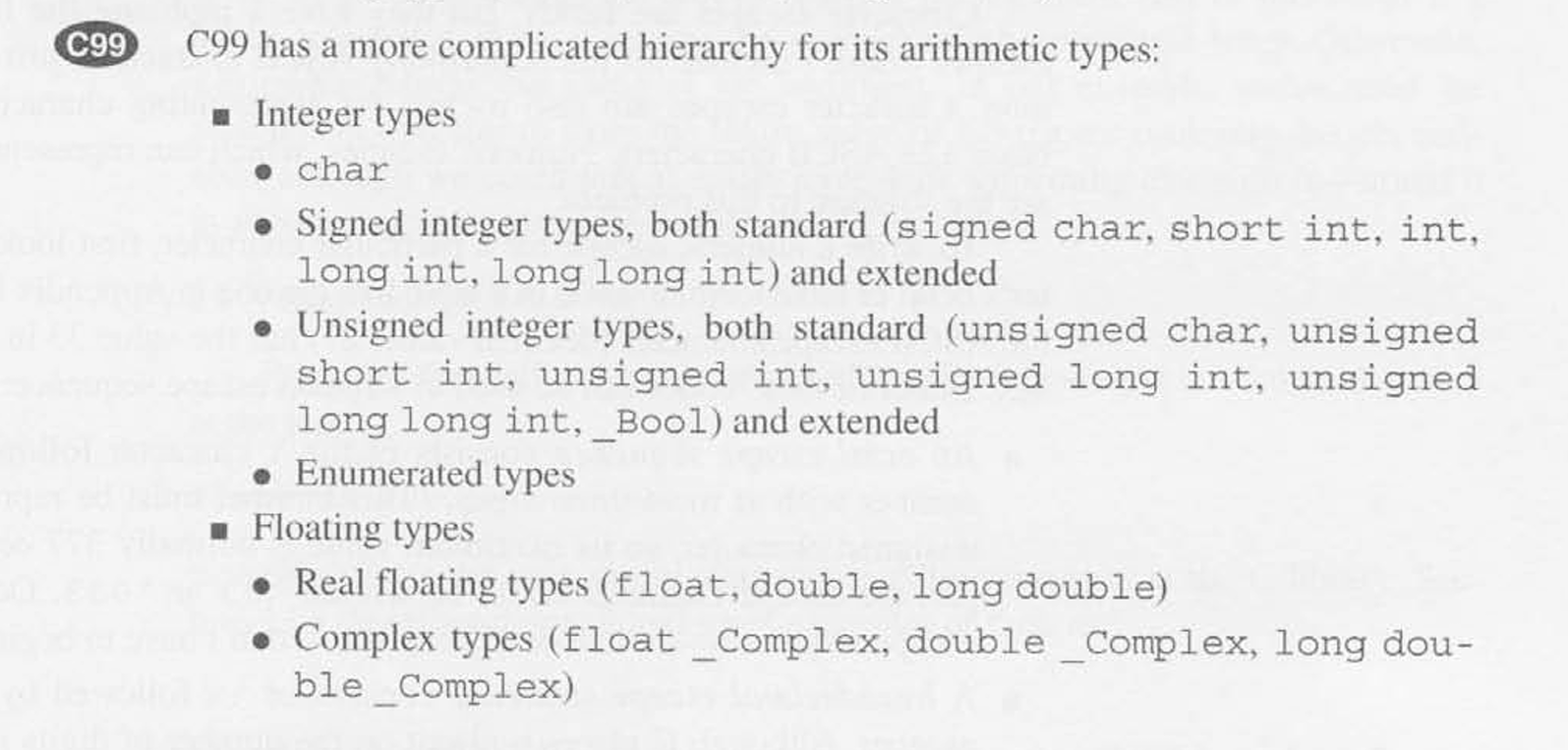
long long int and unsigned long long int is an integer data type provided in C99, are used for very large numbers.

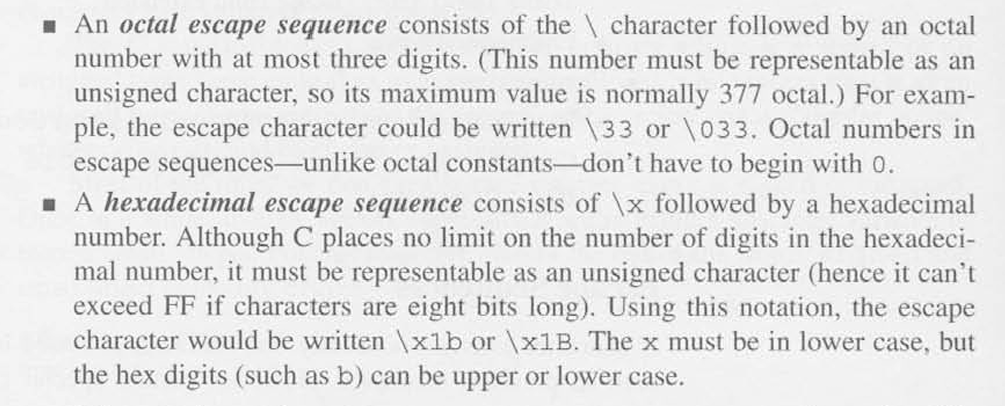
**Integer Constants:** numbers that are just used for representation purposes and are not used to perform read, write and compute operations. They can be represented in decimal, octal and hexadecimal form.  
Decimal constants: contain digits between 0 to 9 and must **not** begin with 0.  
Octal constants: contain digits between 0 to 7 and must begin with 0.  
Hexadecimal constants: contain digits between 0 to 9 and letters from a to f. They always begin with “0x”. The letters can be either uppercase or lowercase or mixed.

**Integer Overflow:** When two numbers of signed int data type are added then the result must be also a signed int and not long or short int, but there are some cases when the result does not agree with the category of int data type and causes an *overflow* resulting in undefined behaviour.When the two numbers of unsigned int data type are added and an overflow takes place then the result will always be zero. Here the result is defined in case of an overflow.

**Conversion Specifiers:**  
unsigned decimal: %u  
unsigned octal: %o  
unsigned hexadecimal: %x  
short: put h in front of d, u, o or x  
long: put l in front of d, u, o or x  
long long: put ll in front of d, u, o or x

**Float Types:**  
float: single precision floating point  
double: double precision floating point  
long double: extended precision floating point  
complex\_types:  
float\_Complex, double\_Complex, long double\_Complex  
By default float constants are saved as double, to save it as float type ‘F’(or f) at last or to save it as long double type ‘L’(or l) at last.  
conversion specifiers:  
for double, type ‘l’ before e, f or g and is required only for scanf and not for printf.  
for long double, type ‘L’ before e, f or g and is required for both scanf and printf.





**ARRAYS:**  
array is a sequence of elements of the same type.

int a[n]; -> defines an array a with n number of elements.

elements can be accessed from the array using indices staring from 0 to n-1 and also values can be assigned.