Basic Types in C language

C provides a standard, minimal set of basic data types. Sometimes these are called "primitive" types. More complex data structures can be built up from these basic types.

Integer Types

The "integral" types in C form a family of integer types. They all behave like integers and can be mixed together and used in similar ways. The differences are due to the different number of bits ("widths") used to implement each type -- the wider types can store a greater ranges of values.

char ASCII character at least 8 bits. Pronounced "car". As a practical matter char is basically always a byte which is 8 bits which is enough to store a single ASCII character. 8 bits provides a signed range of

-128..127 or an unsigned range is 0..255. char is also required to be the "smallest addressable unit" for the machine -- each byte in memory has its own address.

short Small integer -- at least 16 bits which provides a signed range of -32768..32767. Typical size is 16 bits. Not used so much.

int Default integer -- at least 16 bits, with 32 bits being typical. Defined to be the "most comfortable" size for the computer. If you do not really care about the range for an integer variable, declare it int since that is likely to be an appropriate size (16 or 32 bit) which works well for that machine.

long Large integer -- at least 32 bits. Typical size is 32 bits which gives a signed range of about -2 billion ..+2 billion. Some compilers support "long long" for 64 bit ints.

The integer types can be preceded by the qualifier unsigned which disallows representing negative numbers, but doubles the largest positive number representable. For example, a 16 bit implementation of short can store numbers in the range -32768..32767, while unsigned short can store 0..65535. You can think of pointers as being a form of unsigned long on a machine with 4 byte pointers. In my opinion, it's best to avoid using unsigned unless you really need to. It tends to cause more misunderstandings and problems than it is worth.

char Constants

A char constant is written with single quotes (') like 'A' or 'z'. The char constant 'A' is really just a synonym for the ordinary integer value 65 which is the ASCII value for uppercase 'A'. There are special case char constants, such as '\t' for tab, for characters which are not convenient to type on a keyboard.

'A' uppercase 'A' character
'\n' newline character
'\t' tab character
'\0' the "null" character -- integer value 0 (different from the char digit '0')
'\012' the character with value 12 in octal, which is decimal 10

int Constants

Numbers in the source code such as 234 default to type int. They may be followed by an 'L' (upper or lower case) to designate that the constant should be a long such as 42L. An integer constant can be written with a leading 0x to indicate that it is expressed in hexadecimal -- 0x10 is way of expressing the number 16. Similarly, a constant may be written in octal by preceding it with "0" -- 012 is a way of expressing the number 10.

Type Combination and Promotion

The integral types may be mixed together in arithmetic expressions since they are all basically just integers with variation in their width. For example, char and int can be combined in arithmetic expressions such as ('b' + 5). How does the compiler deal with the different widths present in such an expression? In such a case, the compiler "promotes" the smaller type (char) to be the same size as the larger type (int) before combining the values. Promotions are determined at compile time based purely on the types of the values in the expressions. Promotions do not lose information -- they always convert from a type to compatible, larger type to avoid losing information.

Floating point Types

'l' (long double). Single precision equates to about 6 digits of float
Single precision floating point number typical size: 32 bits double
Double precision floating point number typical size: 64 bits long double

Possibly even bigger floating point number (somewhat obscure) Constants in the source code such as 3.14 default to type double unless the are suffixed with an 'f' (float) or precision and double is about 15 digits of precision. Most C programs use double for their computations. The main reason to use float is to save memory if many numbers need to be stored. The main thing to remember about floating point numbers is that they are inexact. For example, what is the value of the following double expression?

```
(1.0/3.0 + 1.0/3.0 + 1.0/3.0) // is this equal to 1.0 exactly?
```

The sum may or may not be 1.0 exactly, and it may vary from one type of machine to another. For this reason, you should never compare floating numbers to eachother for equality (==) -- use inequality (<) comparisons instead. Realize that a correct C program run on different computers may produce slightly different outputs in the rightmost digits of its floating point computations.

Comments

Comments in C are enclosed by slash/star pairs: `/* .. comments .. */` which may cross multiple lines. C++ introduced a form of comment started by two slashes and extending to the end of the line: // comment until the line end The // comment form is so handy that many C compilers now also support it, although it is not technically part of the C language.

Along with well-chosen function names, comments are an important part of well written code. Comments should not just repeat what the code says. Comments should describe what the code accomplishes which is much more interesting than a translation of what each statement does. Comments should also narrate what is tricky or non-obvious about a section of code.

Variables

As in most languages, a variable declaration reserves and names an area in memory at run time to hold a value of particular type. Syntactically, C puts the type first followed by the name of the variable. The following declares an int variable named "num" and the 2nd line stores the value 42 into num.

```
int num;
num = 42;
```

A variable corresponds to an area of memory which can store a value of the given type. Making a drawing is an excellent way to think about the variables in a program. Draw each variable as box with the current value inside the box. This may seem like a "beginner" technique, but when I'm buried in some horribly complex programming problem, I invariably resort to making a drawing to help think the problem through.

Variables, such as num, do not have their memory cleared or set in any way when they are allocated at run time.Variables start with random values, and it is up to the program to set them to something sensible before depending on their values.

Names in C are case sensitive so "x" and "X" refer to different variables. Names can contain digits and underscores (_), but may not begin with a digit. Multiple variables can be declared after the type by separating them with commas. C is a classical "compile time" language -- the names of the variables, their types, and their implementations are all flushed out by the compiler at compile time.