2.4. Keywords and identifiers

After covering the underlying alphabet, we can look at more interesting elements of C. The most obvious of the language elements are *keywords* and *identifiers*; their forms are identical (although their meanings are different).

2.4.1. Keywords

C keeps a small set of *keywords* for its own use. These keywords cannot be used as identifiers in the program — a common restriction with modern languages. Where users of Old C may be surprised is in the introduction of some new keywords; if those names were used as identifiers in previous programs, then the programs will have to be changed. It will be easy to spot, because it will provoke your compiler into telling you about invalid names for things. Here is the list of keywords used in Standard C; you will notice that none of them use upper-case letters.

`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`

Table 2.3. Keywords

The new keywords that are likely to surprise old programmers are: `const`, `signed`, `void` and `volatile` (although `void` has been around for a while).

Eagle eyed readers may have noticed that some implementations of C used to use the keywords `entry`, `asm`, and `fortran`. These are not part of the Standard, and few will mourn them.

2.4.2. Identifiers

Identifier is the fancy term used to mean ‘name’. In C, identifiers are used to refer to a number of things: we've already seen them used to name variables and functions. They are also used to give names to some things we haven't seen yet, amongst which are *labels* and the ‘tags’ of *structures*, *unions*, and *enums*.

The rules for the construction of identifiers are simple: you may use the 52 upper and lower case alphabetic characters, the 10 digits and finally the underscore ‘_’, which is considered to be an alphabetic character for this purpose. The only restriction is the usual one; identifiers **must** start with an alphabetic character.

Although there is no restriction on the length of identifiers in the Standard, this is a point that needs a bit of explanation. In Old C, as in Standard C, there has **never** been any restriction on the length of identifiers. The problem is that there was never any guarantee that more than a certain number of characters would be checked when names were compared for equality—in Old C this was eight characters, in Standard C this has changed to 31.

So, practically speaking, the new limit is 31 characters—although identifiers **may** be longer, they must differ in the first 31 characters if you want to be sure that your programs are portable. The Standard allows for implementations to support longer names if they wish to, so if you do use longer names, make sure that you don't rely on the checking stopping at 31.

One of the most controversial parts of the Standard is the length of *external identifiers*. External identifiers are the ones that have to be visible outside the current source code file. Typical examples of these would be library routines or functions which have to be called from several different source files.

The Standard chose to stay with the IS restrictions on these external names: they are not guaranteed to be different unless they differ from each other in the first six characters. Worse than that, upper and lower case letters may be treated the same!

The reason for this is a pragmatic one: the way that most C compilation systems work is to use operating system specific tools to bind library functions into a C program. These tools are outside the control of the C compiler writer, so the Standard has to impose realistic limits that are likely to be possible to meet. There is nothing to prevent any specific implementation from giving better limits than these, but for maximum portability the six monocase characters must be all that you expect. The Standard warns that it views both the use of only one case and any restriction on the length of external names to less than 31 characters as obsolescent features. A later standard may insist that the restrictions are lifted; let's hope that it is soon.

Previous section | Chapter contents | Next section