TECHNICAL STUFF

Even more keyword madness!

Keywords are worth noting because their use is restricted or reserved. For example, you cannot think up your own function and name it `short`. That's because `short` is a keyword, reserved only for its specific purpose in the core C language. That's one way the keywords are special.

In addition to the 32 keywords shown in Table 3-1 are these two depreciated C language keywords:

> fortran
>
> entry

C once had these keywords, but no longer. Still, I would avoid using them in your programs. (That's what "depreciated" means.)

Also, the C++ language has a hoard of reserved words. If you plan to study C++, include these words in your do-not-use, reserved C language vocabulary:

asm	false	private	throw
bool	friend	protected	true
catch	inline	public	try
class	mutable	reinterpret_cast	typeid
const_cast	namespace	static_cast	using
delete	new	template	virtual
dynamic_cast	operator	this	

It's better to know these words now and not use them than to use one (such as `new` or `friend`) and run into trouble later when you eventually find out how to use C++.

In addition to grammar, languages require rules, exceptions, jots and tittles, and all sorts of fun and havoc. Programming languages are similar to spoken language in that they have various parts and lots of rules.

- ✐ The keywords can also be referred to as *reserved words*.

- ✐ Note that all keywords are lowercase. This sentence is always true for C: Keywords, as well as the names of functions, are lowercase. C is case sensitive, so there is a difference between `return`, `Return`, and `RETURN`.

- ✐ You are never required to memorize the 32 keywords.

- ✐ In fact, of the 32 keywords, you may end up using only half on a regular basis.

- ✐ Some keywords are real words! Others are abbreviations or combinations of two or more words. Still others are cryptograms of the programmers' girlfriends' names.

- ✐ Each of the keywords has its own set of problems. You don't just use the keyword `else`, for example; you must use it *in context*.