and >, not the characters [and]. In contrast, the string "??(??)" has length two, because the compiler replaces the trigraph ??(by the character [and the trigraph ??) by the character].

Digraphs are more limited than trigraphs. First, as we've seen, digraphs are of no use inside a string literal or character constant; trigraphs are still needed in these situations. Second, digraphs don't solve the problem of providing alternate representations for the characters \, ^, |, and ~. The <iso646.h> header, described next, helps with this problem.

C99

The `<iso646.h>` Header: Alternative Spellings

The <iso646.h> header is quite simple. It contains nothing but the definitions of the eleven macros shown in Table 25.10. Each macro represents a C operator that contains one of the characters &, |, ~, !, or ^, making it possible to use the operators listed in the table even when these characters are absent from the keyboard.

Table 25.10
Macros Defined in
<iso646.h>

Macro	*Value*
and	&&
and_eq	&=
bitand	&
bitor	|
compl	~
not	!
not_eq	!=
or	||
or_eq	|=
xor	^
xor_eq	^=

The name of the header comes from ISO/IEC 646, an older standard for an ASCII-like character set. This standard allows for "national variants," in which countries substitute local characters for certain ASCII characters, thereby causing the problem that digraphs and the <iso646.h> header are trying to solve.

25.4 Universal Character Names (C99)

Section 25.2 discussed the Universal Character Set (UCS), which is closely related to Unicode. C99 provides a special feature, ***universal character names,*** that allows us to use UCS characters in the source code of a program.

A universal character name resembles an escape sequence. However, unlike ordinary escape sequences, which can appear only in character constants and string literals, universal character names may also be used in identifiers. This feature allows programmers to use their native languages when defining names for variables, functions, and the like.