In C89, there are no standard pragmas—they're all implementation-defined.

C99 C99 has three standard pragmas, all of which use STDC as the first token following #pragma. These pragmas are FP_CONTRACT (covered in Section 23.4), CX_LIMITED_RANGE (Section 27.4), and FENV_ACCESS (Section 27.6).

C99 The _Pragma Operator

C99 introduces the _Pragma operator, which is used in conjunction with the #pragma directive. A _Pragma expression has the form

_Pragma expression

_Pragma (*string-literal*)

When it encounters such an expression, the preprocessor "destringizes" the string literal (yes, that's the term used in the C99 standard!) by removing the double quotes around the string and replacing the escape sequences \" and \\ by the characters " and \, respectively. The result is a series of tokens, which are then treated as though they appear in a #pragma directive. For example, writing

```
_Pragma("data(heap_size => 1000, stack_size => 2000)")
```

is the same as writing

```
#pragma data(heap_size => 1000, stack_size => 2000)
```

The _Pragma operator lets us work around a limitation of the preprocessor: the fact that a preprocessing directive can't generate another directive. _Pragma, however, is an operator, not a directive, and can therefore appear in a macro definition. This makes it possible for a macro expansion to leave behind a #pragma directive.

Let's look at an example from the GCC manual. The following macro uses the _Pragma operator:

```
#define DO_PRAGMA(x) _Pragma(#x)
```

The macro would be invoked as follows:

```
DO_PRAGMA(GCC dependency "parse.y")
```

After expansion, the result will be

```
#pragma GCC dependency "parse.y"
```

which is one of the pragmas supported by GCC. (It issues a warning if the date of the specified file—parse.y in this example—is more recent than the date of the current file—the one being compiled.) Note that the argument to the call of DO_PRAGMA is a series of tokens. The # operator in the definition of DO_PRAGMA causes the tokens to be stringized into "GCC dependency \"parse.y\"": this string is then passed to the _Pragma operator, which destringizes it, producing a #pragma directive containing the original tokens.