Table 27.8 `<fenv.h>` Macros

Name	*Value*	*Description*

`FE_DIVBYZERO`
`FE_INEXACT`
`FE_INVALID`
`FE_OVERFLOW`
`FE_UNDERFLOW`
	Integer constant expressions whose bits do not overlap	Defined only if the corresponding floating-point exception is supported by the implementation. An implementation may define additional macros that represent floating-point exceptions.

`FE_ALL_EXCEPT`	See description	Bitwise *or* of all floating-point exception macros defined by the implementation. Has the value 0 if no such macros are defined.

`FE_DOWNWARD`
`FE_TONEAREST`
`FE_TOWARDZERO`
`FE_UPWARD`
	Integer constant expressions with distinct nonnegative values	Defined only if the corresponding rounding direction can be retrieved and set via the `fegetround` and `fesetround` functions. An implementation may define additional macros that represent rounding directions.

`FE_DFL_ENV`	A value of type `const fenv_t *`	Represents the default (program start-up) floating-point environment. An implementation may define additional macros that represent floating-point environments.

The `FENV_ACCESS` Pragma

#pragma directive ➤ *14.5*

The `<fenv.h>` header provides a pragma named `FENV_ACCESS` that's used to notify the compiler of a program's intention to use the functions provided by this header. Knowing which portions of a program will use the capabilities of `<fenv.h>` is important for the compiler, because some common optimizations can't be performed if control modes don't have their customary settings or may change during program execution.

The `FENV_ACCESS` pragma has the following appearance:

```
#pragma STDC FENV_ACCESS on-off-switch
```

where *on-off-switch* is either `ON`, `OFF`, or `DEFAULT`. If the pragma is used with the value `ON`, it informs the compiler that the program might test floating-point status flags or alter a floating-point control mode. The value `OFF` indicates that flags won't be tested and default control modes are in effect. The meaning of `DEFAULT` is implementation-defined; it represents either `ON` or `OFF`.

The duration of the `FENV_ACCESS` pragma depends on where it's used in a program. When it appears at the top level of a source file, outside any external declarations, it remains in effect until the next `FENV_ACCESS` pragma or the end of the file. The only other place that an `FENV_ACCESS` pragma might appear is at the beginning of a compound statement (possibly the body of a function); in that case, the pragma remains in effect until the next `FENV_ACCESS` pragma (even one inside a nested compound statement) or the end of the compound statement. At the end of a compound statement, the state of the switch returns to its value before the compound statement was entered.

It's the programmer's responsibility to use the `FENV_ACCESS` pragma to indicate regions of a program in which low-level access to floating-point hardware