*fdim* The `fdim` function computes the positive difference of `x` and `y`:

$$\begin{cases} x - y & \text{if } x > y \\ +0 & \text{if } x \leq y \end{cases}$$

*fmax* *fmin* The `fmax` function returns the larger of its two arguments. `fmin` returns the value of the smaller argument.

## Floating Multiply-Add

```
double fma(double x, double y, double z);
float fmaf(float x, float y, float z);
long double fmal(long double x, long double y,
                 long double z);
```

*fma* The `fma` function multiplies its first two arguments, then adds the third argument. In other words, we could replace the statement

```
a = b * c + d;
```

with

```
a = fma(b, c, d);
```

This function was added to C99 because some newer CPUs have a "fused multiply-add" instruction that both multiplies and adds. Calling `fma` tells the compiler to use this instruction (if available), which can be faster than performing separate multiply and add instructions. Moreover, the fused multiply-add instruction performs only one rounding operation, not two, so it may produce a more accurate result. It's particularly useful for algorithms that perform a series of multiplications and additions, such as the algorithms for finding the dot product of two vectors or multiplying two matrices.

To determine whether calling the `fma` function is a good idea, a C99 program can test whether the `FP_FAST_FMA` macro is defined. If it is, then calling `fma` should be faster than—or at least as fast as—performing separate multiply and add operations. The `FP_FAST_FMAF` and `FP_FAST_FMAL` macros play the same role for the `fmaf` and `fmal` functions, respectively.

Performing a combined multiply and add is an example of what the C99 standard calls "contraction," where two or more mathematical operations are combined and performed as a single operation. As we saw with the `fma` function, contraction often leads to better speed and greater accuracy. However, programmers may wish to control whether contraction is done automatically (as opposed to calls of `fma`, which are explicit requests for contraction), since contraction can lead to slightly different results. In extreme cases, contraction can avoid a float-point exception that would otherwise be raised.