parameter list for the function, where the C++ type of each successive parameter is determined by the corresponding type in `Ts`. The resulting construct is a function parameter pack.

Conceptually, a single variadic template function declaration can be thought of as a multitude of similar declarations with zero, one, etc., parameters fashioned after the variadic declaration:

```
template <typename... Ts> void f1c(const Ts&...);
    // variadic, any number of arguments, any types, all by const &

void f1c();
    // pseudo-equivalent for variadic f1c called with 0 arguments

template <typename T0> void f1c(const T0&);
    // pseudo-equivalent for variadic f1c called with 1 argument

template <typename T0, typename T1>
void f1c(const T0&, const T1&);
    // pseudo-equivalent for variadic f1c called with 2 arguments

template <typename T0, typename T1, typename T2>
void f1c(const T0&, const T1&, const T2&);
    // pseudo-equivalent for variadic f1c called with 3 arguments

// ...                            (and so on ad infinitum)
```

A good intuitive model would be that the pack expansion in the function parameter list is like an "elastic" list of parameter declarations that expands or shrinks appropriately. Note that the types in a pack expansion may all be different from one another, but the qualifiers (**`const`** and/or **`volatile`**) and declarator operators (i.e., pointer `*`, reference `&`, **`rvalue reference`** `&&`, and array `[]`) specified in the function parameter pack are applied to each argument.

A variadic function template can take additional template parameters as well as additional function parameters:

```
template <typename... Ts> void f2a(int, Ts...);
    // one int followed by zero or more arbitrary arguments by value

template <typename T, typename... Ts> void f2b(T, const Ts&...);
    // first by value, zero or more by const &

template <typename T, typename U, typename... Ts> void f2c(T, const U&, Ts...);
    // first by value, second by const &, zero or more by value
```

There are restrictions on such declarations; see *The Rule of Greedy Matching* on page 896 and *The Rule of Fair Matching* on page 898.