Variadic functions

Variadic functions are functions (e.g. `std::printf`) which take a variable number of arguments.

To declare a variadic function, an ellipsis is used as the last parameter, e.g. `int printf(const char* format, ...);`. See Variadic arguments for additional detail on the syntax, automatic argument conversions and the alternatives.

To access the variadic arguments from the function body, the following library facilities are provided:

Defined in header `<cstdarg>`

`va_start`	enables access to variadic function arguments (function macro)
`va_arg`	accesses the next variadic function argument (function macro)
`va_copy` (C++11)	makes a copy of the variadic function arguments (function macro)
`va_end`	ends traversal of the variadic function arguments (function macro)
`va_list`	holds the information needed by va_start, va_arg, va_end, and va_copy (typedef)

Example

Run this code

```
#include <iostream>
#include <cstdarg>

void simple_printf(const char* fmt...)
{
    va_list args;
    va_start(args, fmt);

    while (*fmt != '\0') {
        if (*fmt == 'd') {
            int i = va_arg(args, int);
            std::cout << i << '\n';
        } else if (*fmt == 'c') {
            // note automatic conversion to integral type
            int c = va_arg(args, int);
            std::cout << static_cast<char>(c) << '\n';
        } else if (*fmt == 'f') {
            double d = va_arg(args, double);
            std::cout << d << '\n';
        }
        ++fmt;
    }

    va_end(args);
}

int main()
{
    simple_printf("dcff", 3, 'a', 1.999, 42.5);
}
```

Output:

```
3
a
1.999
42.5
```

See also

C documentation for **`Variadic functions`**