from_chars

This lesson mainly elaborates on from_chars function types.

WE'LL COVER THE FOLLOWING

- Integral type functions
- Floating point version:
- chars_format
  - `from_chars_result`

`from_chars` is a set of overloaded functions: for integral types and floating point types.

Integral type functions

```
std::from_chars_result from_chars(const char* first,
                                  const char* last,
                                  TYPE &value,
                                  int base = 10);
```

Where `TYPE` expands to all available signed and unsigned integer types and char. base can be a number ranging from **2 to 36**.

Floating point version:

```
std::from_chars_result from_chars(const char* first,
                                  const char* last, FLOAT_TYPE& value,
                                  chars_format fmt = chars_format::general);
```

`FLOAT_TYPE` expands to `float`, `double` or `long double`.

chars_format

chars_format is an `enum` with the following values:

```
enum class chars_format {
    scientific = /*unspecified*/,
    fixed = /*unspecified*/,
    hex = /*unspecified*/,
    general = fixed | scientific
};
```

It's a bit-mask type, that's why the values for enums are implementation-specific. By default, the format is set to be general so the input string can use "normal" floating-point format with scientific form as well.

The return value in all of those functions (for integers and floats) is `from_chars_result` :

```
struct from_chars_result {
    const char* ptr;
    std::errc ec;
};
```

`from_chars_result`

`from_chars_result` holds valuable information about the conversion process.

Here's the summary:

- On **Success** `from_chars_result::ptr` points at the first character not matching the pattern, or has the value equal to `last` if all characters match and `from_chars_result::ec` is value-initialized.
- On **Invalid conversion** `from_chars_result::ptr` equals `first` and `from_chars_result::ec` equals `std::errc::invalid_argument`. `value` is unmodified.
- On **Out of range** - The number is too large to fit into the value type. `from_chars_result::ec` equals `std::errc::result_out_of_range` and `from_chars_result::ptr` points at the first character not matching the pattern. `value` is unmodified.

Now that you've learned about `from_chars`, next lesson will discuss its two types; floating and integral.