std::strtof, std::strtod, std::strtold

Defined in header <cstdlib>

```
float       strtof( const char* str, char** str_end );      (since C++11)
double      strtod( const char* str, char** str_end );
long double strtold( const char* str, char** str_end );     (since C++11)
```

Interprets a floating point value in a byte string pointed to by `str`.

Function discards any whitespace characters (as determined by `std::isspace()`) until first non-whitespace character is found. Then it takes as many characters as possible to form a valid floating-point representation and converts them to a floating-point value. The valid floating-point value can be one of the following:

- decimal floating-point expression. It consists of the following parts:
  - (optional) plus or minus sign
  - nonempty sequence of decimal digits optionally containing decimal-point character (as determined by the current C locale) (defines significand)
  - (optional) **e** or **E** followed with optional minus or plus sign and nonempty sequence of decimal digits (defines exponent to base 10)
- hexadecimal floating-point expression. It consists of the following parts:
  - (optional) plus or minus sign
  - **0x** or **0X**
  - nonempty sequence of hexadecimal digits optionally containing a decimal-point character (as determined by the current C locale) (defines significand)
  - (optional) **p** or **P** followed with optional minus or plus sign and nonempty sequence of decimal digits (defines exponent to base 2)
- infinity expression. It consists of the following parts: (since C++11)
  - (optional) plus or minus sign
  - **INF** or **INFINITY** ignoring case
- not-a-number expression. It consists of the following parts:
  - (optional) plus or minus sign
  - **NAN** or **NAN(***char_sequence***)** ignoring case of the **NAN** part. *char_sequence* can only contain digits, Latin letters, and underscores. The result is a quiet NaN floating-point value.
- any other expression that may be accepted by the currently installed C locale

The functions sets the pointer pointed to by `str_end` to point to the character past the last character interpreted. If `str_end` is a null pointer, it is ignored.

Parameters

str	–	pointer to the null-terminated byte string to be interpreted
str_end	–	pointer to a pointer to character.

Return value

Floating point value corresponding to the contents of `str` on success. If the converted value falls out of range of corresponding return type, range error occurs and HUGE_VAL, HUGE_VALF or HUGE_VALL is returned. If no conversion can be performed, `0` is returned and *str_end is set to str.

Example

Run this code

```
#include <iostream>
#include <string>
#include <cerrno>
#include <cstdlib>
#include <clocale>
```

```
int main()
{
    const char* p = "111.11 -2.22 0X1.BC70A3D70A3D7P+6 -Inf 1.18973e+4932zzz";
    char* end;
    std::cout << "Parsing \"" << p << "\":\n";
    for (double f = std::strtod(p, &end); p != end; f = std::strtod(p, &end))
    {
        std::cout << "  '" << std::string(p, end-p) << "' -> ";
        p = end;
        if (errno == ERANGE){
            std::cout << "range error, got ";
            errno = 0;
        }
        std::cout << f << '\n';
    }

    if (std::setlocale(LC_NUMERIC, "de_DE.utf8")) {
        std::cout << "With de_DE.utf8 locale:\n";
        std::cout << "  '123.45' -> " << std::strtod("123.45", 0) << '\n';
        std::cout << "  '123,45' -> " << std::strtod("123,45", 0) << '\n';
    }
}
```

Possible output:

```
Parsing "111.11 -2.22 0X1.BC70A3D70A3D7P+6 -Inf 1.18973e+4932zzz":
  '111.11' -> 111.11
  ' -2.22' -> -2.22
  ' 0X1.BC70A3D70A3D7P+6' -> 111.11
  ' -Inf' -> -inf
  ' 1.18973e+4932' -> range error, got inf
With de_DE.utf8 locale:
  '123.45' -> 123
  '123,45' -> 123.45
```

See also

atof	converts a byte string to a floating point value (function)
wcstof **wcstod** **wcstold**	converts a wide string to a floating point value (function)
from_chars (C++17)	converts a character sequence to an integer or floating-point value (function)
C documentation for **strtof, strtod, strtold**	