std::atof

Defined in header <cstdlib>

```
double atof( const char *str );
```

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

Parameters

str - pointer to the null-terminated byte string to be interpreted

Return value

`double` value corresponding to the contents of `str` on success. If the converted value falls out of range of the return type, the return value is undefined. If no conversion can be performed, `0.0` is returned.

Example

Run this code

```
#include <cstdlib>
#include <iostream>

int main()
{
    std::cout << std::atof("0.0000000123") << '\n'
              << std::atof("0.012")        << '\n'
              << std::atof("15e16")        << '\n'
              << std::atof("-0x1afp-2")    << '\n'
              << std::atof("inF")          << '\n'
              << std::atof("Nan")          << '\n'
              << std::atof("invalid")      << '\n';
}
```

Output:

```
1.23e-08
0.012
1.5e+17
-107.75
inf
nan
0
```

See also

stof (C++11) **stod** (C++11) **stold** (C++11)	converts a string to a floating point value (function)
strtof **strtod** **strtold**	converts a byte string to a floating point value (function)
from_chars (C++17)	converts a character sequence to an integer or floating-point value (function)
atoi **atol** **atoll** (C++11)	converts a byte string to an integer value (function)
C **documentation** for **atof**	