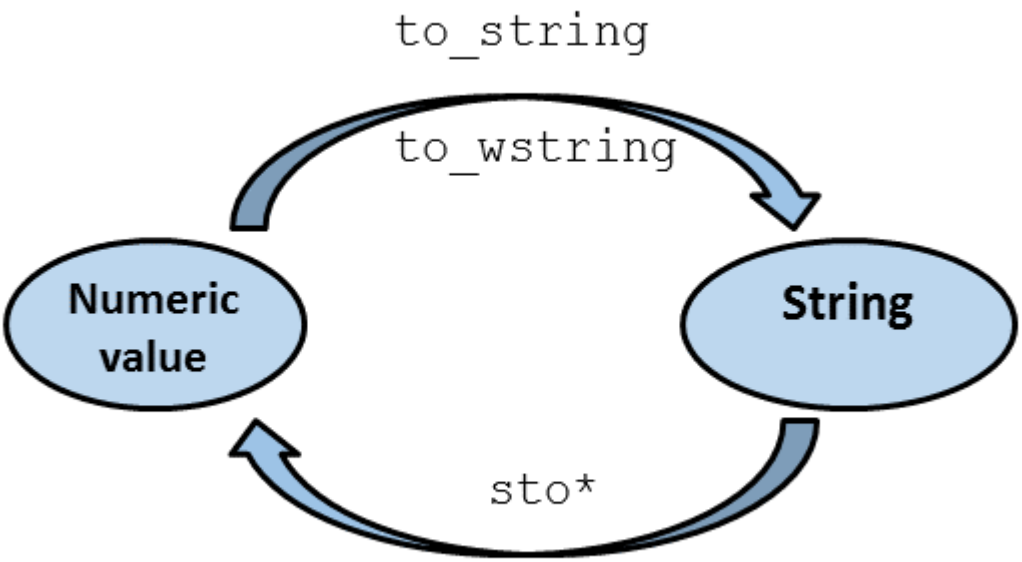
Numeric Conversions

Apart from conversion to C string, a string can also be converted to a float.



You can convert with `std::to_string(val)` and `std::to_wstring(val)` numbers or floating point numbers to the corresponding `std::string` or `std::wstring`. For the opposite direction for the numbers or floating point numbers, you have the function family of the `sto`* functions. All functions need the header `<string>`.

> 🗝️ **Read sto * as string to**
> The seven ways to convert a string into a natural or floating point number follow a simple pattern. All functions start with `sto` and add further characters, denoting the type to which the strings should be converted to. E.g. `stol` stands for string to `long` or `stod` for string to `double`.

The `sto` functions all have the same interface. The example shows it for the type `long`.

```
std::stol(str, idx= nullptr, base= 10)
```

The function takes a string and determines the `long` representation to the base `base`. `stol` ignores leading spaces and optionally returns the index of the first invalid character in `idx`. By default, the base is 10. Valid values for the base are 0 and 2 until 36. If you use base 0 the compiler automatically determines the type based on the format of the string. If the base is bigger than 10 the compiler encodes them in the characters `a` until `z`. The representation is analogous to the representation of hexadecimal numbers.

The table gives an overview of all functions.

Method	Description
`std::to_string(val)`	Converts `val` into a `std::string`.
`std::to_wstring(val)`	Converts `val` into a `std::wstring`.
`std::stoi(str)`	Returns an `int` value.
`std::stol(str)`	Returns a `long` value.
`std::stoll(str)`	Returns a `long long` value.
`std::stoul(str)`	Returns an `unsigned long` value.
`std::stoull(str)`	Returns an `unsigned long long` value.
`std::stof(str)`	Returns a `float` value.
`std::stod(str)`	Returns a `double` value.
`std::stold(str)`	Returns an `long double` value.

Numeric conversion of strings

> **ℹ Where is the stou function?**
> In case you're curious, the C++ `sto` functions are thin wrappers around the C `strto*` functions, but there is no `strtou` function in C. Therefore C++ has no `stou` function.

The functions throw a `std::invalid_argument` exception if the conversion is not possible. If the determined value is too big for the destination type you get a `std::out_of_range` exception.

```
#include <iostream>
#include <limits>
#include <string>

int main(){

  //std::cout << std::endl;

  std::cout << "to_string, to_wstring" << std::endl;

  std::string maxLongLongString=std::to_string(std::numeric_limits<long long>::max());
  std::wstring maxLongLongWstring=std::to_wstring(std::numeric_limits<long long>::max());

  std::cout << std::numeric_limits<long long>::max() << std::endl;
  std::cout << maxLongLongString << std::endl;
  std::wcout <<  maxLongLongWstring <<  std::endl;

  std::cout << std::endl;

  std::cout << "ato* " << std::endl;

  std::string str("10010101");

  std::cout << std::stoi(str) << std::endl;
  std::cout << std::stoi(str, nullptr, 16) << std::endl;
  std::cout << std::stoi(str, nullptr, 8) << std::endl;
  std::cout << std::stoi(str, nullptr, 2) << std::endl;

  std::cout << std::endl;

  std::size_t idx;
  std::cout << std::stod("   3.5 km", &idx) << std::endl;
  std::cout << "Not numeric char at position " << idx << "." << std::endl;

  std::cout << std::endl;

  try{
    std::cout << std::stoi("   3.5 km") << std::endl;
    std::cout << std::stoi("   3.5 km", nullptr, 2) << std::endl;

  }
  catch (const std::exception& e){
```

```
    std::cerr <<  e.what() << std::endl;
  }

  std::cout << std::endl;

}
```

Numeric conversion (Expected Error)