RandAnything

0.0.1

Generated by Doxygen 1.8.9.1

Mon Nov 9 2015 19:30:03

Contents

Chapter 1

RandAnything

Because *pseudo*-**random** should be **easy**.

This is a single-file (_RandAnything.h_) library that defines an easy-to-use pseudo-random number generator `RandAnything`. `RandAnything` can generate random values within a range for almost any numerical type, and even strings!

Using the Header File in your Project

All you need to do to use `RandAnything` is to place a copy of the header file _RandAnything.h_ into your project (or a system include folder like `/usr/local/include/`) and then `#include` it in your program.

Easy.

Generating Random Numeric Values

Now, you are two steps away from random value bliss. First, declare an instance of the `RandAnything` generator, using a template argument corresponding to the kind of value you want to generate. For example, if you want to generate integer values to simulate a 6-sided die roll, you do this: "‘cpp RandAnything<int> die_roller; // Generator for integer values "‘ Second, you just call the generator object as if it were a function, passing the lower- and upper- bounds for the value you want to generate: "‘cpp int d6 = die_roller(1, 6); // Simulate rolling a 6-sided die. "‘ That's it! The 1 represents the *low* end of the range of possible values, and the 6 represents the *high* end of the range. A uniform random number in the range [1,6] will be returned.

`RandAnything` will allow you to generate any kind of numeric value you want by following those two steps.

Range Note: The range of values generated for integer-based values is a *closed range* [low, high], but floating-point-based values are produced in an *open range* [low, high) (the high value is never actually produced, but values can get vanishingly close, according to the precision available in the type itself).

Generating `std::string` Values

`RandAnything` can also generate randomized strings of type `std::string`! The process is nearly the same, but there are a few more options.

For one thing, you can choose the *alphabet* (set of characters) from which the string is chosen. By default it will consist of all *non-whitespace, printable* characters (that is, characters that aren't whitespace and will result in a visible character on the screen). But you can customize the alphabet easily (more on that later).

Also, you can choose to *either* generate strings with a *variable length* (whose length is in a range specified by [low, high] similar to the numeric ranges shown above), or you can generate strings with a fixed length.

Example Code: “‘cpp // Set up the generator for strings: RandAnything<std::string> string_gen; // Generate a string with a length between 4 and 12 characters: std::string random_str_1 = string_gen(4,12);

// Generate a string whose length is exactly 8 characters: std::string random_str_2 = string_gen(8);

// Generate a string of exactly 8 characters while limiting the // alphabet to "01" (producing something that looks like a // binary number): std::string random_str_3 = string_gen(8, "01");

// Generate a string with between 3 and 9 characters from the // alphabet "abc123" std::string random_str_4 = string_↩ gen(3, 9, "abc123"); “‘

Bonus: Easy Alphabets

The `std::string` version of `RandAnything` knows about several special alphabets that will come in handy as well. The following methods are available:

- `alphabet_alphaLowerCase()` - lowercase letters
- `alphabet_alphaUpperCase()` - uppercase letters
- `alphabet_alphaAllCase()` - lowercase and uppercase letters
- `alphabet_numeric()` - numeric digits 0 through 9
- `alphabet_alphaNumeric()` - lowercase and uppercase letters and numeric digits
- `alphabet_punctuation()` - punctuation characters (non-whitespace printable characters that aren't letters or numbers)
- `alphabet_printable()` - all printable non-whitespace characters
- `alphabet_hexadecimal()` - characters that are valid for hexadecimal digits (`0` through `9` and `a` through `f`)

Examples: “‘cpp // Set up the generator for strings: RandAnything<std::string> string_gen;

// Generate an alphanumeric password: std::string pass = string_gen(32, string_gen.alphabet_alphaNumeric());

// Generate a string of lowercase letters between 4 and 16 characters long: std::string lc = string_gen(4, 16, string_↩ gen.alphabet_alphaLowerCase()); “‘

Fiddly Details

`RandAnything` is designed to give a good trade-off between quality pseudo-random values and performance when generating lots of values. Although some performance was sacrificed for ease-of-use, but it should still be plenty fast enough for most applications.

Seeding

When you declare the instance of the `RandAnything` object, it will perform an initial setup operation involving obtaining a source of randomness using a source of *true random entropy* if such a source is available. (The mechanism used is the `std::random_device`: see `http://en.cppreference.↩ com/w/cpp/numeric/random/random_device`). If a source of nondeterministic entropy is not available, then you end up with a deterministic seeding function.

Generation

Random number generation depends on a Mersenne Twister algorithm (specifically `std::mt19937`: see http://en.cppreference.com/w/cpp/numeric/random/mersenne_twister_engine). This produces *pseudo-random* values based on a (in most cases) true-random seed. The result is that the part of the process that takes significant time (the true-random seeding) occurs only at instantiation, and on every call a comparatively *fast* pseudo-random generation occurs.

License and Copyright

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

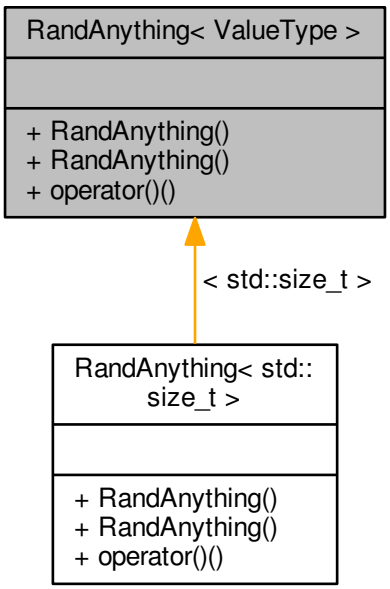
Class Documentation

4.1 RandAnything< ValueType > Class Template Reference

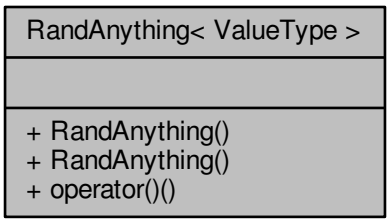
Generate a random value of any numeric type or std::string.

`#include <RandAnything.h>`

Inheritance diagram for RandAnything< ValueType >:



Collaboration diagram for RandAnything< ValueType >:



Public Member Functions

- RandAnything ()
- RandAnything (unsigned int seed)
- ValueType operator() (const ValueType &low, const ValueType &high)

4.1.1 Detailed Description

template<typename ValueType>class RandAnything< ValueType >

Generate a random value of any numeric type or std::string.

Generate (almost) any type of uniform random value in a range [low,high] (for integral values) or [low, high) (for floating-point values). Just instantiate the class with whatever type you want as the template argument, then use it as a function where the arguments are the lower and upper bounds of the range for the resulting random value. To generate `std::string` values, RandAnything<std::string> specialization.

Template Parameters

ValueType	Type of value that should be generated. Supports integral types, Real-number types, and `std::string`.

Definition at line 122 of file RandAnything.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 template<typename ValueType > RandAnything< ValueType >::RandAnything ( )

constructs the random number generator and prepares it for use; seeding is automatic and uses a non-deterministic seed (if available on the system)

Template Parameters

ValueType	Type of value that should be generated. Supports integral types, Real-number types, and `std::string`.

Definition at line 139 of file RandAnything.h.

4.1.2.2 template<typename ValueType > RandAnything< ValueType >::RandAnything (unsigned int *seed*)

constructs the random number generator and prepares it for use given an explicit seed

Parameters

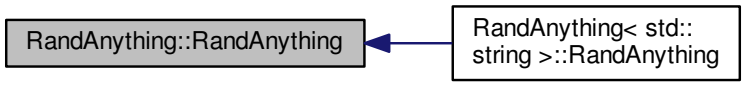
seed	A seeding value. An entire state sequence is generated from this value using a linear random generator.

Template Parameters

ValueType	Type of value that should be generated. Supports integral types, Real-number types, and `std::string`.

Definition at line 149 of file RandAnything.h.

Here is the caller graph for this function:



4.1.3 Member Function Documentation

4.1.3.1 template<typename ValueType> ValueType RandAnything< ValueType >::operator() (const ValueType & *low*, const ValueType & *high*)

Generate random value in range [low,high].

Parameters

low	smallest value that can be generated
high	largest value that can be generated

Template Parameters

ValueType	Type of value that should be generated. Supports integral types, Real-number types, and `std::string`.

Returns

a (uniform) random number in the range [low, high]

Definition at line 161 of file RandAnything.h.

The documentation for this class was generated from the following file:

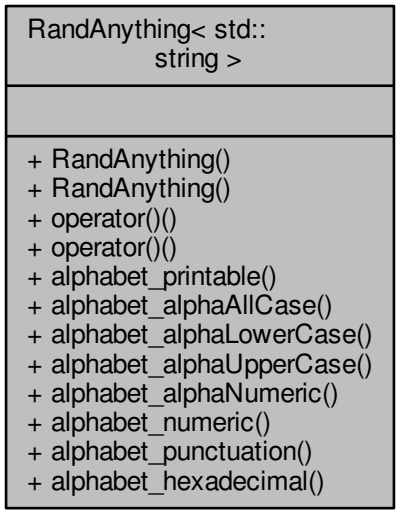
- RandAnything.h

4.2 RandAnything< std::string > Class Template Reference

RandAnything specialization for std::string generation.

`#include <RandAnything.h>`

Collaboration diagram for RandAnything< std::string >:



Public Member Functions

- RandAnything ()
- RandAnything (unsigned int seed)
- std::string operator() (std::size_t length, std::string alphabet="")

  generate a random `std::string` of a specific length from a chosen alphabet
- std::string operator() (std::size_t min_length, std::size_t max_length, std::string alphabet="")

  generate a random `std::string` in a range of lengths from a chosen alphabet
- std::string alphabet_printable () const

  generates the alphabet of all printable (non-whitespace) characters
- std::string alphabet_alphaAllCase () const

  generates the alphabet of all alphabetical characters (upper- and lower-case)
- std::string alphabet_alphaLowerCase () const

  generates the alphabet of all lowercase alphabetical characters

- std::string alphabet_alphaUpperCase () const

  generates the alphabet of all uppercase alphabetical characters

- std::string alphabet_alphaNumeric () const

  generates the alphabet of all alphabetical (upper- and lower-case) and numeric digits

- std::string alphabet_numeric () const

  generates the alphabet of all numeric characters

- std::string alphabet_punctuation () const

  generates the alphabet of all punctuation and symbol characters (all non-whitespace printable characters that are not alphabetical or numeric)

- std::string alphabet_hexadecimal () const

  generates the alphabet of all hexadecimal digits [0,f]

4.2.1 Detailed Description

template<>class RandAnything< std::string >

RandAnything specialization for std::string generation.

Generates std::strings with either a fixed length or with a range of lengths given an alphabet of characters to choose from (or using all printable characters). This class also exposes methods to generate several useful alphabets.

Template Parameters

[description]	

Definition at line 208 of file RandAnything.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 RandAnything< std::string >::RandAnything ( ) [inline],[default]

Constructs an instance of RandAnything for generating std::string values, using a non-deterministic seed (if available)

4.2.2.2 RandAnything< std::string >::RandAnything (unsigned int *seed*) [inline]

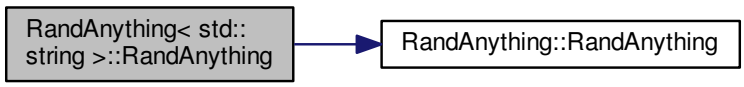
Constructs an instance of RandAnything for generating std::string values, using an explicit seed given by the `seed` parameter.

Parameters

seed	seed value that determines the sequence of values produced by the generator

Definition at line 238 of file RandAnything.h.

Here is the call graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 std::string RandAnything< std::string >::alphabet_alphaAllCase ( ) const

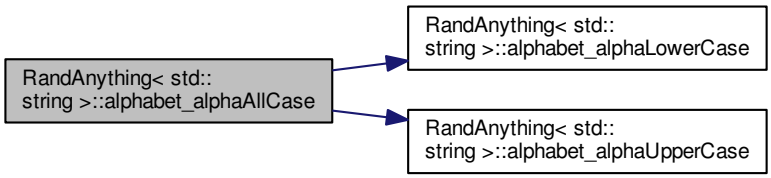
generates the alphabet of all alphabetical characters (upper- and lower-case)

Returns

a `std::string` consisting of all alphabetical characters

Definition at line 307 of file RandAnything.h.

Here is the call graph for this function:



4.2.3.2 std::string RandAnything< std::string >::alphabet_alphaLowerCase ( ) const

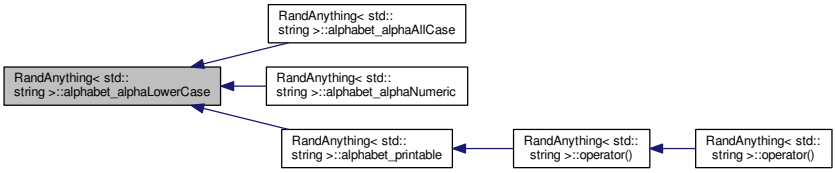
generates the alphabet of all lowercase alphabetical characters

Returns

a `std::string` consisting of all lowercase alphabetical characters

Definition at line 277 of file RandAnything.h.

Here is the caller graph for this function:



4.2.3.3 std::string RandAnything< std::string >::alphabet_alphaNumeric ( ) const

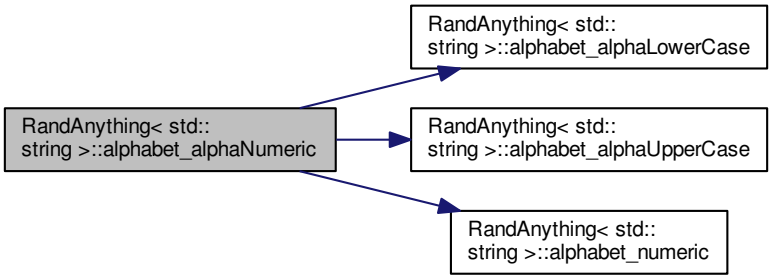
generates the alphabet of all alphabetical (upper- and lower-case) and numeric digits

Returns

a `std::string` consisting of all alphabetical characters and digits

Definition at line 314 of file RandAnything.h.

Here is the call graph for this function:



4.2.3.4 std::string RandAnything< std::string >::alphabet_alphaUpperCase ( ) const

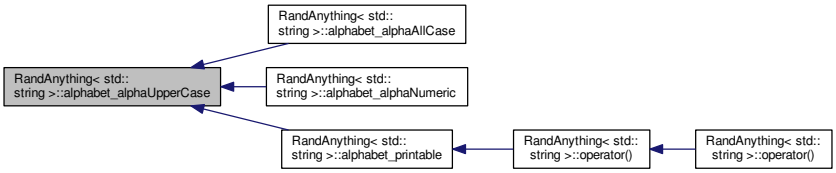
generates the alphabet of all uppercase alphabetical characters

Returns

a `std::string` consisting of all uppercase alphabetical characters

Definition at line 287 of file RandAnything.h.

Here is the caller graph for this function:



4.2.3.5 std::string RandAnything< std::string >::alphabet_hexadecimal ( ) const

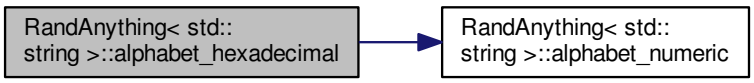
generates the alphabet of all hexadecimal digits [0,f]

Returns

a `std::string` consisting of all hexadecimal digits [0,f]

Definition at line 345 of file RandAnything.h.

Here is the call graph for this function:



4.2.3.6 std::string RandAnything< std::string >::alphabet_numeric ( ) const

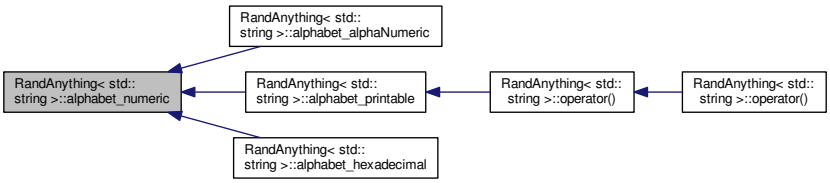
generates the alphabet of all numeric characters

Returns

a `std::string` consisting of all numeric characters

Definition at line 297 of file RandAnything.h.

Here is the caller graph for this function:



4.2.3.7 std::string RandAnything< std::string >::alphabet_printable ( ) const

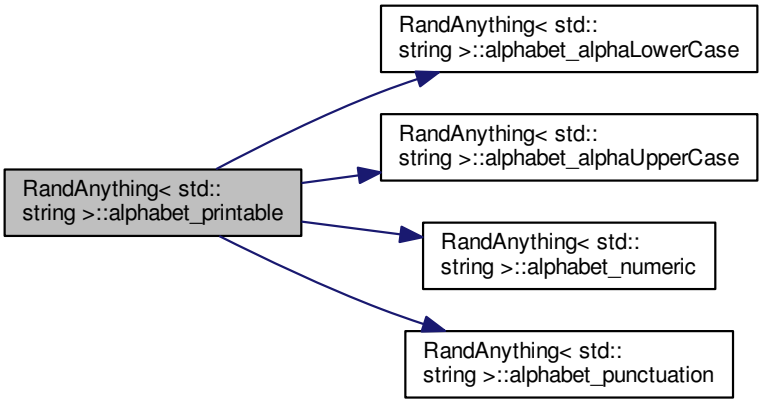
generates the alphabet of all printable (non-whitespace) characters

Returns

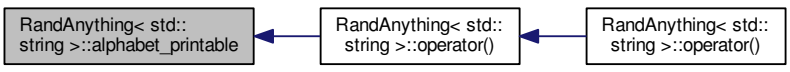
a `std::string` consisting of all printable (non-whitespace) characters

Definition at line 338 of file RandAnything.h.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.8 std::string RandAnything< std::string >::alphabet_punctuation ( ) const

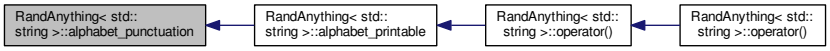
generates the alphabet of all punctuation and symbol characters (all non-whitespace printable characters that are not alphabetical or numeric)

Returns

a `std::string` consisting of all punctuation and symbol printable characters

Definition at line 322 of file RandAnything.h.

Here is the caller graph for this function:



4.2.3.9 std::string RandAnything< std::string >::operator() (std::size_t *length*, std::string *alphabet* = " ") [inline]

generate a random `std::string` of a specific length from a chosen alphabet

Generates a `std::string` of characters containing characters chosen at random from `alphabet` (uniform choice, with replacement). The length of the generated string is given by `length`.

Parameters

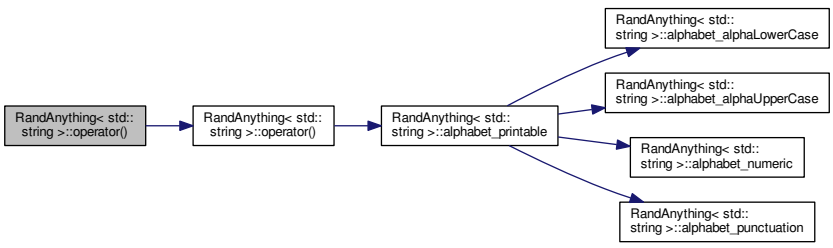
length	length of generated string
alphabet	set of characters that may appear in the generated string

Returns

a random string of characters from `alphabet` whose length is given by `length`

Definition at line 251 of file RandAnything.h.

Here is the call graph for this function:



4.2.3.10 std::string RandAnything< std::string >::operator() (std::size_t *min_length*, std::size_t *max_length*, std::string *alphabet* = " ")

generate a random `std::string` in a range of lengths from a chosen alphabet

Generates a `std::string` of characters containing characters chosen at random from `alphabet` (uniform choice, with replacement). The minimum and maximum possible lengths for the generated string are given by `min_length` and `max_length`, respectively.

Parameters

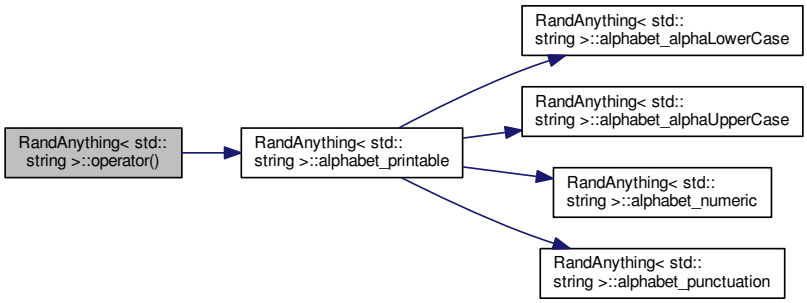
min_length	shortest possible string to generate
max_length	longest possible string to generate
alphabet	set of characters that may appear in the generated string

Returns

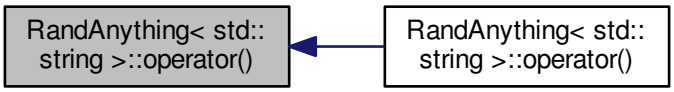
a random string of characters from `alphabet` whose length is in the range [`min_length`, `max_length`]

Definition at line 268 of file RandAnything.h.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- RandAnything.h

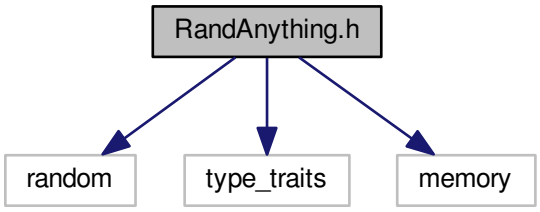
Chapter 5

File Documentation

5.1 RandAnything.h File Reference

```
#include <random>
#include <type_traits>
#include <memory>
```

Include dependency graph for RandAnything.h:



Classes

- class RandAnything< ValueType >

  Generate a random value of any numeric type or std::string.
- class RandAnything< std::string >

  RandAnything specialization for std::string generation.

5.1.1 Detailed Description

Defines a class RandAnything that will allow you to quickly generate a quality pseudo-random value of (almost) any standard type without worrying about STL type names or doing a lot of setup.

Copyright

Index