# dcmt-cpp

*Release 1.0.0*

**Takahiro Ueda**

**Oct 02, 2020**

# CONTENTS

template<class **UIntType**, std::size_t **WordSize**, std::size_t **p**>
**class** dcmt::**dcmt_param**

Set of parameters for Mersenne Twister pseudo-random number generators.

**Template Parameters**

- `UIntType`: unsigned integral type for generators
- `WordSize`: word size for pseudo-random numbers
- `p`: Mersenne prime exponent

## Public Types

**using result_type** = *UIntType*

Integral type for generators.

## Public Functions

**dcmt_param**(*result_type id* = *default_id*, *result_type seed* = *default_seed*)

Searches for a set of parameters for pseudo-random number generators.

**Parameters**

- `id`: to generate the characteristic polynomial. Must be <= 65535
- `seed`: random seed to generate the characteristic polynomial

## Public Static Attributes

**constexpr** std::size_t **word_size** = *WordSize*

Word size that determines the range of generated numbers.

**constexpr** std::size_t **prime_exponent** = *p*

Mersenne prime exponent.

**constexpr** *result_type* **default_id** = 0

Default id to search for the characteristic polynomial.

**constexpr** *result_type* **default_seed** = 4172

Default random seed to search for the characteristic polynomial.

template<class **UIntType**, std::size_t **WordSize**, std::size_t **p**>
**class** dcmt::**dcmt_engine**

Mersenne Twister pseudo-random number generator engine.

**Template Parameters**

- `UIntType`: unsigned integral type for the engine
- `WordSize`: word size for the generated numbers
- `p`: Mersenne prime exponent

### Public Types

**using param_type** = *dcmt_param*<*UIntType*, *WordSize*, *p*>
Parameter type for the engine.

**using result_type** = *UIntType*
Integral type generated by the engine.

### Public Functions

**dcmt_engine**(*result_type seed* = *default_seed*)
Constructs the engine.

**Parameters**

- `seed`: random seed

**dcmt_engine**(**const** *param_type &param*, *result_type seed* = *default_seed*)
Constructs the engine.

**Parameters**

- `param`: set of parameters
- `seed`: random seed

**dcmt_engine**(*param_type &&param*, *result_type seed* = *default_seed*)
Constructs the engine.

**Parameters**

- `param`: set of parameters
- `seed`: random seed

void **seed**(*result_type value* = *default_seed*)
Reinitializes the engine by a new seed value.

**Parameters**

- `value`: random seed

*result_type* **operator()**()
Returns the next pseudo-random number.

**Return** generated value

void **discard**(unsigned long long *z*)
Advances the internal state.

**Parameters**

- `z`: number of advances

### Public Static Functions

**constexpr** *result_type* **min**()
Returns the minimum value potentially generated by the engine, which is 0.

**Return** minimum value

**constexpr** *result_type* **max**()
Returns the maximum value potentially generated by the engine.

**Return** maximum value

### Public Static Attributes

**constexpr** std::size_t **word_size** = *WordSize*
Word size that determines the range of numbers generated by the engine.

**constexpr** std::size_t **prime_exponent** = *p*
Mersenne prime exponent.

**constexpr** *result_type* **default_seed** = 3241
Default random seed.

### Friends

**friend** bool **operator==**(**const** *dcmt_engine* &*a*, **const** *dcmt_engine* &*b*)
Compares two engines.

**Return** `true` if the engines are equivalent including their internal states, `false` otherwise

**Parameters**

- `a`: first engine
- `b`: second engine

**friend** bool **operator!=**(**const** *dcmt_engine* &*a*, **const** *dcmt_engine* &*b*)
Compares two engines.

**Return** `true` if the engines are not equivalent including their internal states, `false` otherwise

**Parameters**

- `a`: first engine
- `b`: second engine

template<class **CharT**, class **Traits**>
**friend** std::basic_ostream<*CharT*, *Traits*> &**operator<<**(std::basic_ostream<*CharT*, *Traits*> &*os*, **const** *dcmt_engine* &*e*)
Serializes the state of the given engine into a stream.

**Return** `os`

**Parameters**

- `os`: output stream

- `e`: engine to be serialized

template<class **CharT**, class **Traits**>
**friend** std::basic_istream<*CharT*, *Traits*> &**operator>>**(std::basic_istream<*CharT*, *Traits*> &*is*, *dcmt_engine* &*e*)

Deserializes the state of the given engine from a stream.

**Return** `is`

**Parameters**

- `is`: input stream
- `e`: engine to be deserialized

**using** `dcmt::`**dcmt521** = *dcmt_engine*<std::uint_fast32_t, 32, 521>

32-bit Mersenne Twister pseudo-random number generator engine with a period 2^521-1.

CHAPTER

ONE

# CHANGELOG

## 1.1 1.0.0 - 2020-10-02

### 1.1.1 Added

- First release.

CHAPTER

TWO

# DCMT-CPP

A C++11 wrapper library for Dynamic Creator of Mersenne Twisters.

## 2.1 Example

```
#include <iostream>
#include <dcmt/dcmt.h>

int main() {
  // Create 32-bit Mersenne Twister pseudo-random number generator
  // engines with a period 2^521-1. Their characteristic polynomials are
  // determined with id = 0, 1, 999 and generated sequences are highly
  // independent. The generator engines are initialized with seed =
  // 1234, 4567, 8901, respectively.
  dcmt::dcmt521 rng1{dcmt::dcmt521::param_type{0}, 1234};
  dcmt::dcmt521 rng2{dcmt::dcmt521::param_type{1}, 4567};
  dcmt::dcmt521 rng3{dcmt::dcmt521::param_type{999}, 8901};

  // Generate a number by using each generator engine.
  std::cout << rng1() << std::endl;
  std::cout << rng2() << std::endl;
  std::cout << rng3() << std::endl;

  return 0;
}
```

### 2.1.1 CMake (3.14 or later)

```
add_subdirectory(dcmt-cpp)
target_link_libraries(your_app PRIVATE dcmt)
```

## 2.2 Development

```
# Install prerequisites (minimal).
brew install cmake gcc git

# Install prerequisites (including optional ones).
brew install cmake doxygen gcc git lcov llvm pre-commit

# Install pre-commit hooks.
pre-commit install
pre-commit install --hook-type commit-msg

# Set CPM source cache.
export CPM_SOURCE_CACHE=$HOME/.cache/CPM  # for Linux
export CPM_SOURCE_CACHE=$HOME/Library/Caches/CPM  # for macOS

# Run linters.
pre-commit run --all-files

# Tests.
cmake -S . -B build/debug -DCMAKE_BUILD_TYPE=Debug
cmake --build build/debug --target check

# Benchmarking.
cmake -S . -B build/release -DCMAKE_BUILD_TYPE=Release -DBUILD_BENCHMARKING=ON
NANOBENCH_SUPPRESS_WARNINGS=1 cmake --build build/release --target bench

# Documents.
cmake -S . -B build/docs -DBUILD_TESTING=OFF
cmake --build build/docs --target doc

# Code coverage.
cmake -S . -B build/coverage -DCMAKE_BUILD_TYPE=Debug -DUSE_CODE_COVERAGE=ON
lcov -z -d build/coverage
cmake --build build/coverage --target check
lcov -c -d build/coverage -o build/coverage/coverage.info
lcov -r build/coverage/coverage.info '*/c++/*' -o build/coverage/coverage.info
lcov -r build/coverage/coverage.info '*/lib/*' -o build/coverage/coverage.info
lcov -r build/coverage/coverage.info '*/doctest/*' -o build/coverage/coverage.info
lcov -r build/coverage/coverage.info '*/tests/*' -o build/coverage/coverage.info
genhtml -o build/coverage/html build/coverage/coverage.info

# Compiler sanitizers.
cmake -S . -B build/sanitizer -DCMAKE_BUILD_TYPE=Debug -DUSE_SANITIZER=ON
cmake --build build/sanitizer --target check

# Clang-Tidy.
cmake -S . -B build/clang-tidy -DCMAKE_BUILD_TYPE=Debug -DUSE_CLANG_TIDY=ON
cmake --build build/clang-tidy --target check
```

## 2.3 License

MIT

# INDEX

## D