# TDDD38 - Extra lecture

## The random header

Eric Elfving

Department of Computer and Information Science
Linköping University

LINKÖPING UNIVERSITY

This library creates random values with the help of Generators and Distributions.
Generators generate uniformly distrubuted integer values, Distributions transforms sequences of numbers generated by a Generator into a given statistical distribution (such as normal, uniform and poisson)

# Generators

Most generators are random number engines - using algorithms to generate a specific sequence of pseudo-random numbers from a given seed value. There is one exception - `random_device`. `random_device` is a non-deterministic number generator that uses stochastic processes to generate it's values, if supported.

# Generators

`random_device` is usually slow so using a random engine is prefered. You can use `random_device` to generate the seed value (or some other value such as `system_clock::now`).

```
mt19937 gen {random_device{}() };
mt19937 gen2 {4};
```

`mt19937` stands for a "mersenne twister" engine with a period of $2^{19937} - 1$

# Generators

Generators can be initialized (seeded) with an integer value or a object with a `generate` member function with the same signature as `std::seed_seq::generate` (or with a `seed_seq` object).

```
std::string str;
std::cout << "Please, enter a seed: ";
std::getline(std::cin, str);
std::seed_seq seed {str.begin(), str.end()};

std::mt19937 gen {seed};
std::cout << "A user seed produced: " << gen() << std::endl;
```

# Distributions

Supports lots of standard distributions (un-)known from your statistics courses such as (all code examples requires some Generator `gen`)

- `uniform_int_distribution`

```
uniform_int_distribution<> die {1, 6};
int val = die(gen); // val has a value in [1, 6]
```

- `normal_distribution`

```
normal_distribution<double> dist;
dist(gen);
```

- `binomial_distribution`

```
binomial_distribution<int> dist(9, 0.4);
dist(gen);
```

# bind

You can make the call a bit easier by using `std::bind`:

```
mt19937 gen { random_device{}() };
uniform_int_distribution<> dist {1,6};
auto die = bind(dist, gen);
int val = die();
```

# Why can't I use rand()?

- Bad range (at least 16 bit $\Rightarrow [0, 32767]$)
- Not uniform (usually uses a linear congruential enginge, very simple algorithm)
- A simple scaling uses modulo: `rand() % 100`... This gives an extra non-uniformity (unless the range of rand() is divisable with our max value)

See the following (great) presentation by Stephan T. Lavavej (STL):
`https://channel9.msdn.com/Events/GoingNative/2013/rand-Considered-Harmful`

www.liu.se

LINKÖPING UNIVERSITY