

# Random Numbers in C++11

## Solutions

- Give an outline description of how to obtain random numbers in modern C++
  - Create a random number engine instance
    - Optionally, this can be seeded using a `random_device` instance
  - Create a suitable distribution instance, giving the range of numbers required as arguments
  - Call the `()` operator of the distribution, passing the engine instance as argument

- What is the purpose of a random number engine?
  - To generate a sequence of random numbers
  - Each time its () operator is called, the next number in the sequence is returned
- Which random number engine is usually the best one to use?
  - mt19937 (Mersenne Twister)

- What is the purpose of a distribution type?
  - Given a sequence of numbers, it will convert them so that they are within a given range and the probability of getting a certain number follows a given statistical distribution
- Which distributions are most useful when generating random numbers?
- What does "uniformly distributed" mean in the context of random numbers?
  - uniform distribution - each number is equally likely

- Write a simple program which
  - a) Prints out ten integers with random values between 0 and 100
  - b) Prints out ten floating point numbers with random values between 0 and 1

- Explain how random\_device generates random numbers
  - random\_device uses information provided by the operating system (hardware device or system entropy)
- How does this differ from a random number engine?
  - random\_device acts as a true random number generator
- Why can we not assume random\_device will work as expected on every system?
  - The operating system may not be able to provide the data
  - In that case, random\_device will fall back to acting as a PRNG

- Why is `random_device` not suitable for generating large quantities of numbers?
  - If the operating system runs out of data, `random_device` has to wait until more data is available
- What is the most common situation where `random_device` is used?
  - For seeding `random_engine`

- Write a simple program which demonstrates the most common way to use `random_device`



- Why should we avoid `default_random_engine`?
  - It is implementation-defined and may be a wrapper around `rand()`
- Why is it necessary to check the documentation before using `random_device`?
  - May act as a PRNG, depending on the environment and the compiler

- Why is it good practice to make engine and distribution instances static?
  - High creation overhead
  - Usually only one instance needed per program
  - Creating a new instance will reset the sequence, starting again from the first number. This may not produce the expected results.