FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Lecture 10

Advanced C++ Standard Library

tuple type

- A tuple is a template similar to pair
- Example
  - tuple<size_t,size_t,size_t> threeD;

Regular expressions

- A regular expression is a way of describing a sequence of characters
- Part of the new C++ standard library found in the regex header file
- Example
  - string pattern(“[^c]ie”);

Random numbers

- The random number engines are function-object classes that define a call operator that takes no argument and returns a random unsigned number
- Example
  - default_random_engine e;
  - cout << e() << endl;

Chrono Library

- A flexible collection of types that track time with varying degrees of precision
- The chrono library defines three main types as well as utility functions and common typedefs
  - clocks
  - time points
  - durations
- Example
  - `auto start = std::chrono::system_clock::now();`
  - `auto end = std::chrono::system_clock::now();`
  - `std::chrono::duration<double> elapsed_seconds = end-start;`
  - `std::time_t end_time = std::chrono::system_clock::to_time_t(end);`
  - `std::cout << "finished computation at " << std::ctime(&end_time) << "elapsed time: " << elapsed_seconds.count() << "s\n";`

Chrono Library

- A flexible collection of types that track time with varying degrees of precision
- The chrono library defines three main types as well as utility functions and common typedefs
  - clocks
  - time points
  - durations
- Example
  - `auto start = std::chrono::system_clock::now();`
  - `auto end = std::chrono::system_clock::now();`
  - `std::chrono::duration<double> elapsed_seconds = end-start;`
  - `std::time_t end_time = std::chrono::system_clock::to_time_t(end);`
  - `std::cout << "finished computation at " << std::ctime(&end_time) << "elapsed time: " << elapsed_seconds.count() << "s\n";`

Other Interesting Library Parts

- std::function
- std::source_location
- std::optional
- std::any
- Complete list:
  - https://en.cppreference.com/w/cpp
- concepts and type traits

How to implement a type trait: SFINAE

- Substitution Failure Is Not An Error (SFINAE)
- This rule applies during overload resolution of function templates: When substituting (replacing by template arguments) the deduced type for the template parameter fails, the specialization is discarded from the overload set instead of causing a compile error.
- Alternatives: static_assert, tag dispatch, enable_if