Utility

- Exceptions
- Random Number Generation
- Chrono
- Type Support
- Variadics
- Sonstige Utilitys

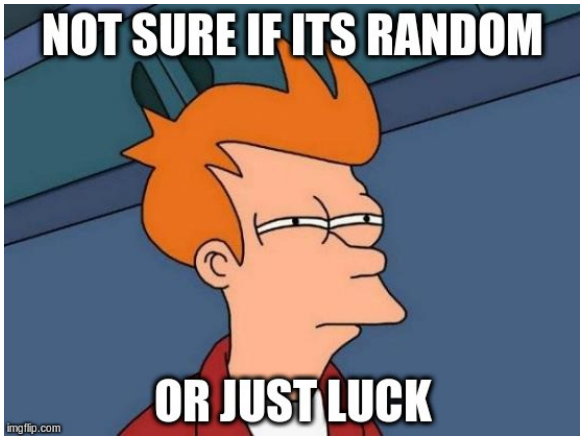
Exceptions

- Grundsätzlich ähnlich wie bei Java
  - Try{} für den block, der Exceptions werfen kann
  - Catch{} um Exceptions zu fangen
  - Throw(..) um eine Exception zu werfen
- Aber:
  - Exceptions müssen nicht von einem bestimmten Typ sein
  - Compiler prüft nicht, ob alle Exceptions behandelt werden
  - Eigene Exceptions Klassen **können** von std::exception abgeleitet werden
  - Mit catch(…) werden alle Exceptions gefangen
- http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#S-errors
- Performance-Overhead ist vorhanden
- https://godbolt.org/g/Rn5q48

Exceptions

- Noexcept Keyword seit C++11
  - Gibt an, dass die Funktion keine Exception schmeißen wird
  - Performance-Vorteil, da Exception Handling komplett ausgeschalten wird
  - Falls doch eine Auftritt, stürzt das Programm ohne Meldung ab
  - http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Re-noexcept

Random Number Generation



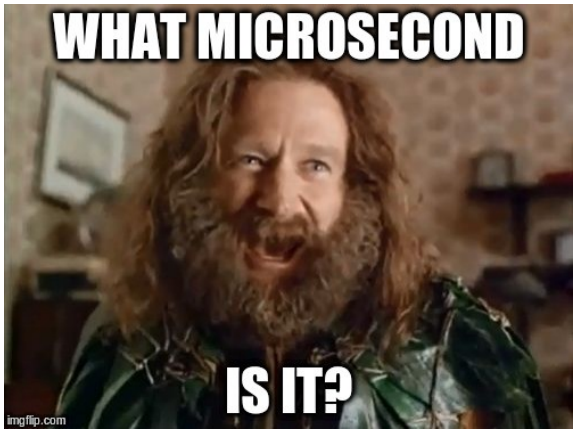
- Probleme bei *rand()*
  - Oft Statistisch Vorhersagbar (Compilerabhängig)
  - Werte Bereich schwer einschränkbar ohne Gleichverteilung zu verlieren
  - Thread safety ist Compilerabhängig
- Seit C++11: <random> Bibliothek

Random Number Generation

- Random Number Generator hat 3 Bestandteile
- Std::random_device
  - Kryptografische sichere unsigend int Zufallswerte
  - Langsame Performance
  - Für Seeds gedacht
- Pseudo Random Engines
  - Vordefinierte Pseudo Zufallszahlengenerator
  - Hohe Performance
  - Vom Standard definiertes Verhalten
  - Empfehlung: std::mt19937
- Random number distributions
  - Garantiert eine bestimmte Verteilung über einen definierbaren Bereich
  - Hat einen Einfluss auf die Performance
  - Empfehlung: uniform_int_distribution
- https://godbolt.org/g/yaY7q8

Chrono

- Seit C++11: <chrono> Bibliothek für Plattformübergreifend Timing
- Drei verschiedene Uhren
  - system_clock: Systeminterne Echtzeituhr
  - steady_clock: Einstellbares Genauigkeit,
  - high_resolution_clock: genaueste Echtzeituhr die das System anbietet
- Mit chrono::duration kann der Datentyp, sowie eine eigene Zeiteinheit erstellt werden
- Mit chrono::duration_cast kann zwischen verschiedenen Zeiteinheiten gecastet werden
- Literale für Standard Zeiteinheiten vorhanden
- https://godbolt.org/g/zkKHXr



Type Support

- nullptr_t als feste Definition als Null Pointer
- size_t als Datentyp für Containergrößen
- numeric_limits
  - Um max und min Werte eines Datentypes zu setzen
  - https://godbolt.org/g/rKjrb3
- Type traits
  - Mehr dazu in der nächsten Vorlesung

Variadics

- Ermöglicht übergabe von beliebigen Argumenten (z.B. Printf)
- Datentyp wird mit „ … “ angegeben
- Mit *va_list* kann man die Informationen abholen
- Mit *va_start* wird der Zugriff auf die Variablen Ermöglicht
- Durch iterieren mit ++ Operator
- Mit *va_end* wird der Aufruf beendet
- https://godbolt.org/g/18u9y7

Swap / Exchange

- Std::swap
  - Vertauscht zwei Variablen
  - Diese müssen den selben Datentyp haben
  - ACHTUNG: z.B. uint16_t ↔ uint32_t geht nicht
- Std::exchange
  - Ersetzt den Wert einer Variable und gibt den ursprünglichen Wert zurück
  - Wert muss moveble sein
  - Auch als atomic_exchange verfübar

bitset

- Einzelne Bitoperationen mit std::bitset
  - Größe wird zur Comilezeit festgelegt
  - Praktische Funktionen wie set, reset, flip, etc.
- https://godbolt.org/g/nJrDCa

Programm Utilities

- Funktionen um Programmablauf zu beeinflussen
  - Std::exit um Programm sauber zu beenden
  - Std::abort um Programm abzubrechen (ohne Cleanup)
- Std::system um Kommandozeilen aufzurufen
- Programmsignale (z.B. SIGSEV)
  - Mit std::signal wird der Signalhandler übergeben
  - Mit std::raise wird ein Signal ausgelöst

Nächstes Mal

Templates und Compile time Polymorphie