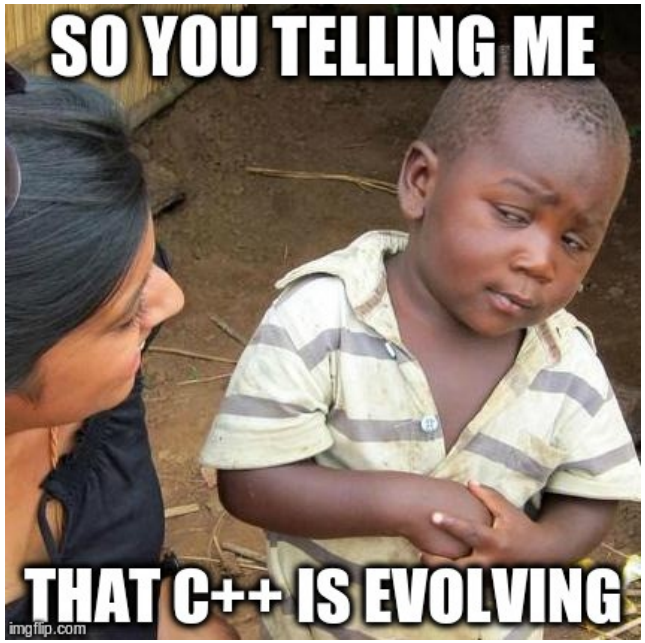
# Ausblick C++17 / C++20

- C++17
  - If/switch initilizer
  - Stuctured Bindings
  - Filesystem
  - Optional /variant/ any
  - constexpr_if
- C++20
  - Concepts
  - Spaceship Operator
- Sonstiges (Working Draft)



# If / switch initializer (17)

- Problem: Werte für if und switch, müssen immer außerhalb des Scopes initialisiert werden und existieren dementsprechend auch außerhalb
- Lösung: if / switch initializer
  - Erlaubt Zuweisung von Werten innerhalb der if / switch Abfrage
- https://godbolt.org/g/F7hnTA

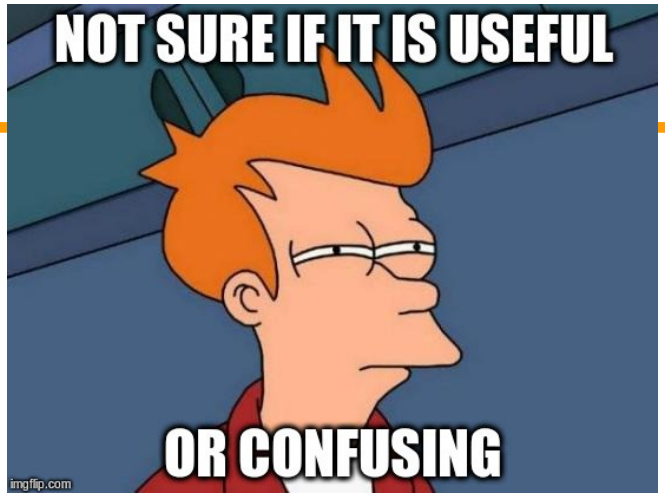
# Structured Bindings (17)

- Problem: Schreibweise für pair / tuple extrem umständlich bei häufiger Verwendung
- Lösung: Structured Bindings
  - Kurzschreibweise für pair / tuple / sturct initialisierung
- https://godbolt.org/g/uHcVPL

# Filesystem (17)

- Einführung eines plattformübergreifenden Dateisystemzugriff
  - Übernommen aus Boost::Filesystem
- Umfangreiche Möglichkeiten Dateien und Verzeichnisse zu manipulieren
- https://godbolt.org/g/NVeKYL

# Optional / any / variant (17)




- 3 neue Datentyp Arten in C++17
  - **Variant** kann mehrere verschiedene Datentypen beinhalten, aber immer nur eines
    - https://godbolt.org/g/5NrYzZ
  - **Any** ist ein Datentyp, der einen beliebigen Datentyp enthalten kann
    - https://godbolt.org/g/QJrVYd
  - **Optional** ist ein Wert der gesetzt sein kann, aber nicht muss
    - https://godbolt.org/g/7diCbE

# Constexpr_if (17)

- Compile Zeit if für constexpr Funktionen
  - Ermöglicht Codeabschnitte für Compilezeit Optimierungen zu schreiben
  - https://godbolt.org/g/5L7kJE

# Concepts (20)

- Concepts ermöglichen vereinfachte Interfaces für Template parameter
- Bisher sind Einschränkungen nur mittels std::enable_if möglich
- Vorallem für Bibliotheken hilfreich
- https://godbolt.org/g/mrjuPn
- http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3351.pdf

# Spaceship Operator (20)

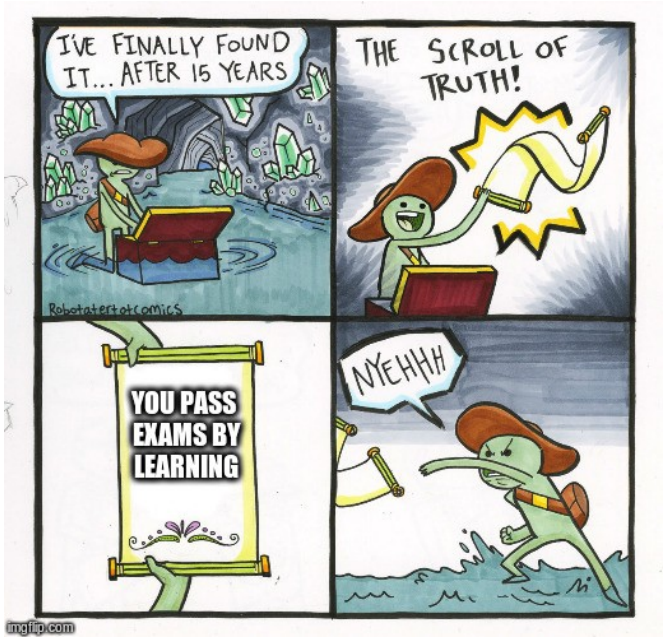

- Offizieller Titel: three-way comparision
- Erlaubt in einer Funktion die  < , >, <= , >= , == , != Operationen zu definieren
- Symbol dafür ist  <=>
- http://open-std.org/JTC1/SC22/WG21/docs/papers/2017/p0515r0.pdf

# Sonstiges

- Networking
  - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4711.pdf
- Reflection
  - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p0194r3.html
- Modules
  - Soll includes ersetzten. Ähnlich wie packages in Java
  - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4681.pdf
- Ranges
  - Vereinfachte Darstellung von Iteratoren
  - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4685.pdf
- Graphics
  - Einfache Grafikbibliothek
  - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p0267r6.pdf
- http://en.cppreference.com/w/cpp/experimental
- http://en.cppreference.com/w/cpp/compiler_support

# ENDE


- Was ihr jetzt (hoffentlich) gelernt habt
  - C++ ist eine extrem performante Sprache
  - Keinen Overhead bei höherer Abstraktion
  - Vertrau dem Compiler
  - Syntax leider oft furchtbar
  - Versucht immer den Modernen Standard zu verwenden
  - Höhere Abstraktion / Templates sind auch im Embedded Bereich verwendbar!!




# Nächstes Mal
# Prüfung