C++11 - Was gibts Neues?

Jan Gampe

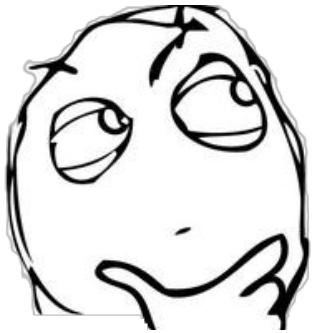
Hochschule RheinMain
Univsersity of Applied Sciences

17. November, 2011

Was ist C++11?

C++11 ist die zweite Neufassung des C++ Standards durch die ISO

- 1998: Erste Fassung (C++98)
- 2003: 1. Neufassung: Fehlerkorrektur (C++03)
- 2007: Library Technical Report (Was sie sich für einen neuen Standard vorstellen könnten)
- 2011: 2. Neufassung (C++11)
- 2012 kann man mit den ersten Implementationen rechnen.



Die gute Nachricht

C++11 holt auf, wo Java und C# schon länger sind und legt noch einen drauf.

Meine Empfehlung

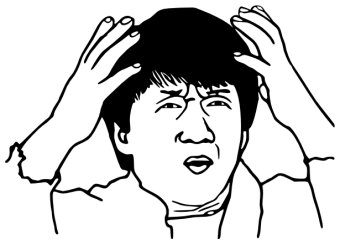
Verlangt, in C++11 zu coden! Lasst es extra kosten, wenn ihr C++98 benutzen wollt.

'C++11 feels like a new language. The pieces just fit together better than they used to.'- **Bjarne Stroustrup**

Intuitive Initialisierungssyntax

Wie legt man eine Variable an?

- int v1 = 7;
- int v2(7);
- int v3 = {7};



C++11 will nun eine Syntax für alles einführen:

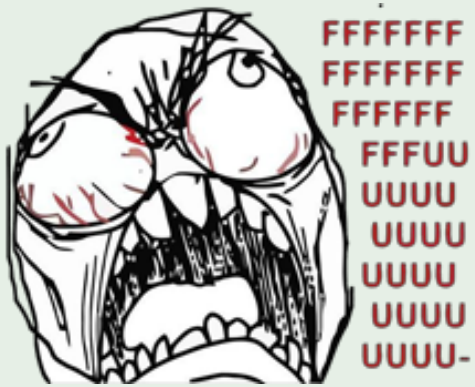
- int v1{7};
- aber auch: int *x = new int{7};

Das gilt auch für Klassen, Strukturen, Arrays, etc.

Intuitive Initialisierungssyntax

C++03

```
vector<int> zahlen;
zahlen.push_back(42);
zahlen.push_back(43);
zahlen.push_back(44);
...
```



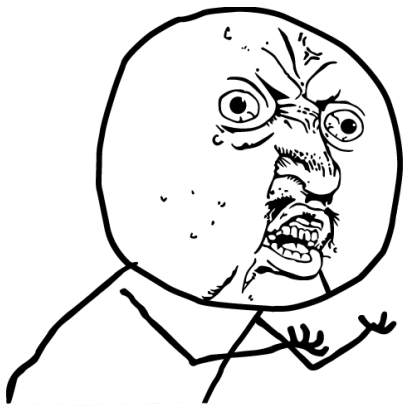
C++11

```
vector<int> zahlen{42,
    43, 44};
```

Multithreading

Multithreading wird nun nativ von C++ unterstützt.

- Thread-lokale Speicherverwaltung
- Atomare Operationen / Zahlen
- Mutexe
- Futures und Promises



Warum erst jetzt?

Für normale PC-Anwendungsentwicklung kommt es spät, aber für die Mikrocontrollerwelt gerade noch rechtzeitig.

Lambda-Ausdrücke

Lambda-Ausdrücke bieten die Möglichkeit, anonyme Funktionen zu schreiben.

Ein Beispiel mit Sort (STL)

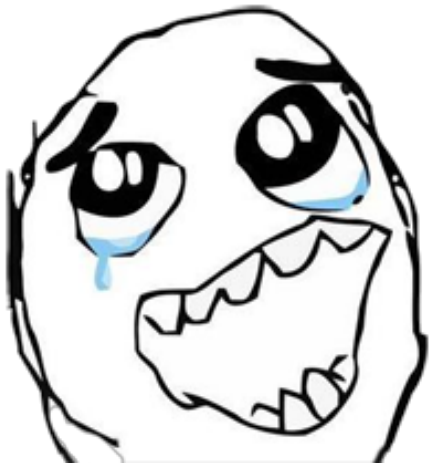
```
std::vector<int> v = {50, -10, 20, -30};
std::sort(v.begin(), v.end(),
      [](int a, int b) { return abs(a)<abs(b); });
```

Variadic Templates

Neben Funktionsparametern kann eine Funktion/Klasse nun auch veränderbar viele Templateargumente erhalten.

Ein N-Tupel in C++

```
template <class ...Types> class tuple;
typedef std::tuple <int, double, long &, const
    char *> test_tuple;
```



For-Each

Überfällig: C++11 kann Range-Based For-Loops (for-each Schleifen).

Beispiel

```
int my_array[5] = {1, 2, 3, 4, 5};
for (int &x : my_array) {
    x *= 2;
}
```

- Funktioniert für Arrays, alle Containerklassen der Standardbibliothek und allem, was ein begin() und end() hat, dessen Rückgabewerte dazwischen iterieren können.

Typinferenz

Wenn für den Compiler klar ist, welchen Datentyp eine Variable hat, braucht man ihn nicht mehr hinschreiben:

auto-Keyword

```
auto irgendwas = 5;
```

- Spart unnötige Schreibarbeit und minimiert Fehler.

Fazit

- Es hat sich noch wesentlich (!) mehr in C++11 getan
- 1338 Seiten C++11 Spec, 700 Seiten C1X Spec
- Mehr dazu bald in meiner Fachseminar-Ausarbeitung

Fragen?

Quellen:

- N1570, aktuelle Arbeitsversion des C1X Entwurfs: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- Offizieller C++11 Standard: ISO/IEC 14882:2011, www.iso.org
- http://www.softwarequalityconnection.com