C++ im Fluss der Zeit

Finn Masurat
Seminar SS2016

Übersicht der Präsentation

- Geschichte
- Kritik
- Die neuen Standards: C++11 und C++14
- Blick in die Zukunft: C++17

Geschichte der Sprache

- Zwischen 1983 und 1985: Einführung von C++

- C++ als Erweiterung von C

- Aus C mit Klassen -> C++

Geschichte der Sprache

- Bjarne Stroustrup - Erfinder der Sprache C++
- Ab 2000er: Starke Zurückdrängung durch Java
- 2005: Ankündigung des neuen Releases C++0x

Kritik an C++

- Verhalten von Sprachkonstrukten nicht definiert
- Fehlerhafter Quellcode wird absurd umgesetzt
- Niemals nur eine Operation ungültig

Kritik an C++

- Komplex und fehleranfällig

- “In C++, it’s harder to shoot yourself in the foot
  but when you do, you blow off your whole leg“

  Bjarne Stroustrup

C++11 und C++14 Features

- System und Bibliotheken Programmierung
- Vereinfachung der Sprache

“auto“

- Wie “var“ in anderen Programmiersprachen

- Vor C++11:
  Ausdruck für den automatischen Stackzuweis

```
int main() {

    auto var1 = 4;
    auto var2 = 3.45;
    auto var3 = “autoauto“;
    auto var4 = new double[10];

}
```

```
int main() {

    list<int> numbers;

    for(int i = 0; i < 10; i++){
        numbers.push_back(i*i);
    }

    for(list<int>::iterator it = numbers.begin();
        it != numbers.end(); it++){

        cout << *it << " ";

    }
```

```
int main() {

    list<int> numbers;

    for(int i = 0; i < 10; i++){
        numbers.push_back(i*i);
    }

    for(auto it = numbers.begin();
        it != numbers.end(); it++){

        cout << *it << " ";

    }
```

```
auto sum(int i, int k){

    return i + k;
}
```

```
int main() {

    int count = 10;
    int& countRef = count;
    auto myAuto = countRef;

    countRef = 11;
    cout << count << " ";

    myAuto = 12;
    cout << count << endl;

}
```

“decltype“

- Fragt den Typ eines Ausdruckes ab

- Oft in der generischen Programmierung verwendet

```
int main() {

    int r = 10;
    decltype(r) dr = 5;

}
```

```
int main() {

    int i = 10;
    int& r = i;
    auto ar = r;
    decltype(r) dr = r;

    const int konstante = 5;
    auto ak = konstante;
    decltype(konstante) = kontante

}
```

“trailing return type“

- Rückgabewert der Funktion nach Parameterübergabe
- Wird zusammen mit auto und decltype() verwendet

```
int sum(int i, int k) {
    return i+k;
}
```

```
auto sum(int i, int k) -> int {
    return i+k;
}
```

```
template<class T>
decltype(i+k) sum(T i, T k){
    return i+k;
}
```

```
template<class T>
auto sum(T i, T k) -> decltype(i+k){
    return i+k;
}
```

```
template<typename T1, typename T2>
auto sum(T1 i, T2 k) -> decltype(i+k){
    return i+k;
}
```

“strongly-typed enums“

- Enum Klasse mit eigenem Scope

```
int main() {

    enum Animals {Bear, Chicken, Cat};
    enum Birds {Eagle, Duck, Chicken};

    enum class Animals {Bear, Chicken, Cat);
    enum class Birds {Eagle, Duck, Chicken};

}
```

```
int main() {

    enum Animals {Bear, Chicken, Cat};
    enum Colors {Red, Blue, Green};

    if (Red < Chicken) {
        …
    }

}
```

“nullpointer“

- Verhindert Mehrdeutigkeit
- Keine Überladung einzelner Methoden

```
void f(int) {
    cout << “integer f“ << endl;
};

void f(char*) {
   cout << “char* f“ << endl;
};

int main() {

f(0);
f(NULL);
f(nullptr);

}
```

```
int main() {

    char* i = nullptr;

    if ( i == 0) {

        cout << 0 << endl;

    }

    i = 0;
    if ( i == nullptr) {

       cout << “nullptr“;

    }
}
```

Weitere neue Feature

- Range Based Loop
- Lambda-Ausdrücke
- constexpr

Blick in die Zukunft - C++17

- Unnötige Funktionen werden entfernt
- Weiterer Fokus auf die generische Programmierung
- Die Sprache attraktiver gestalten

Ende

Quellen

- E. Niebler, D.Gregor, J. Widman, US22/DE9 Revisited:
  Decltype and Call Expressions, Working Paper N3276,
  ISO/IEC JTC1 SC22 WG21 (März 2011).
  URL http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2011/n3276.pdf

- V. Voutilainen, Auto and braced-init-lists, Working Paper N3681,
  ISO/IEC JTC1 SCC22 WG21 (Mai 2013).
  URL http://open-std.org/JTC1/SC22/WG21/docs/papers/2013/n3681.html

- Breymann, Ulrich, Der C++-Programmierer: C++ lernen - professionell anwenden
  Lösungen nutzen
  Carl Hanser Verlag GmbH & Co. KG; Auflage: 3, (Januar 2014)
  ISBN-13: 978-3446438941