# Beyond C

## D, Vala

Alexander Röber

Fachbereich Informatik
Universität Hamburg

3.Juli 2014

# Gliederung

# D - Was ist D? - Überblick

- Objektorientiert!
  - `struct` als value type
  - `class` als reference type

# D - Was ist D? - Überblick

- Objektorientiert!
  - `struct` als value type
  - `class` als reference type
- Programmierfreundlich
  - String als Typ
  - Schlüsselwort `auto`
  - automatisches Speichermanagement

# D - Was ist D? - Überblick

- Objektorientiert!
  - `struct` als value type
  - `class` als reference type
- Programmierfreundlich
  - String als Typ
  - Schlüsselwort `auto`
  - automatisches Speichermanagement
- Performanz
  - Zugriff auf C Bibliotheken
  - Einbauen von Assemblercode

# D - Was ist D? - Weitere Features

- garbage collector
- Vertragsmodell
- assoziative/dynamische Arrays
- `scope`(exit) {...}
- `foreach`(item; list) {...}

- Installation
- Compiler
  - dmd
  - rdmd

# D - Standard HelloWorld

hello.d

```d
import std.stdio;

void main() {
    write("Hello World!");
}
```

```
rdmd hello.d
Hello World!
```

# D - C HelloWorld

hello2.d

```d
import core.stdc.stdio;

void main() {
    printf("Hello World!!!%i!!eins", 1);
}
```

rdmd hello2.d
Hello World!!!1!!eins

# D - Klassen & Referenzen

hello3.d

```d
import std.stdio;

void main() {
    MyClass m = new MyClass();
    int x = 7;
    int y = x;
    m.fortyTwoFy(x);
    write(y, " zweiundvierzigt ist ", x);
}

class MyClass {
    void fortyTwoFy(ref int x) {
        x = 42;
    }
}
```

rdmd hello3.d

7 zweiundvierzigt ist 42

# D - Inline Assembler

```d
import std.stdio;

void main() {
    int x = 6;
    int y = 7;
    int ergebnis = 0;
    version (D_InlineAsm_X86) {
        asm {
            mov EAX, x          ;
            mul EAX, y          ;
            mov ergebnis, EAX   ;
        }
    }
    if(ergebnis != 0) {
        write("Das Produkt aus ", x, " und ", y, " ist ", ergebnis, ".");
    } else {
        write("Das hat nicht funktioniert");
    }
}
```

rdmd hello4.d
Das Produkt aus 6 und 7 ist 42.

# Vala - Was ist das?

- Objektorientiert
- Basiert auf GObject
- Compiler generiert C-Code

# Vala - Warum?

- Standard `using`

# Vala - Warum?

- Standard `using`

```
1 int main() {
2     print("Hello World!");
3     return 0;
4 }
```

# Vala - Warum?

- Standard `using`
- GLib als fester Bestandteil

# Vala - Warum?

- Standard `using`
- GLib als fester Bestandteil
- Viele Möglichkeiten

# Vala - Warum?

- Standard `using`
- GLib als fester Bestandteil
- Viele Möglichkeiten
- Verständliche Syntax

# Vala - Warum nicht?

Warum man nicht Vala einrichten möchte.

Ein Vergleich:

# Vala - Warum nicht?

- Java- / D-Compiler installieren
  - Herunterladen
  - Installieren
  - Fertig!

# Vala - Warum nicht?

- Java- / D-Compiler installieren
  - Herunterladen
  - Installieren
  - Fertig!
- Vala-Compiler installieren
  - C-Compiler installieren
  - Sourcecode herunterladen?
  - kompilieren. . . ?
  - Und GLib nicht vergessen!

# Zusammenfassung

- Euch wurde D vorgestellt und mit ein paar Beispielen näher gebracht.
- Ich habe euch Vala kurz vorgestellt und Vor- und Nachteile aufgezeigt.

# Quellen

http://dlang.org/
http://en.wikipedia.org/wiki/Vala_%28programming_language%29
http://wiki.gnome.org/Projects/Vala