Praxisorientierte Einführung in C++
Lektion: "Typdefinitionen mittels **typedef**"

Christof Elbrechter

Neuroinformatics Group, CITEC

April 24, 2014

Table of Contents

Typedef

- Mit dem `typedef`-Schlüsselwort können leichter lesbare Aliase für komplexe Typen definiert werden.

Syntax

```
typedef AlterTypName NeuerTypName;
```

Beispiel

```
typedef unsigned short int ushort;
typedef float real;
int main(){
  ushort x = 5;
  real r = 7.5;
}
```

Beispiel

- Im Beispiel: besser als `#define ushort unsigned short int`
- Ermöglicht leichten globalen Austausch von Datentypen

my_mathlib.h

```
#ifdef USE_MM_DOUBLE
typedef double mm_real;
#else
typedef float mm_real;
#endif
real add(mm_real a,mm_real b);
real sub(mm_real a,mm_real b);
```

- Durch Übersetzen mit `g++ -DUSE_MM_DOUBLE ...` kann die Präzision der Bibliothek eingestellt werden

Typedef

- Später: Templates, verschachtelte Namespaces etc. $\Rightarrow$ sehr gut für schöneren Code
- Nicht ***immer*** unbedingt vorteilhaft

some_header.h

```
float scalar_product(const Vector &a,const Vector &b);
```

- Wo kann man die Definition von `Vector` finden
- Faustregel: Innerhalb einer Bibliothek sollte die Namensgebung für `typedefs` einem nachvollziehbaren Schema folgen

Typedef für Pointer und Arrays

- Auch Array-Typen und Pointer können mittels `typedef` alternativ benannt werden
- Achtung: `typedef` erstellt nur einen Typalias; der tatsächliche Typ-Bezeichner bleibt gültig

Eigenartige Definitionssyntax für Arrays

```
typedef float Vec3D[3]; // geht, sollte man aber nicht so machen
Vec3D v = {1, 3, 3+3};
```

Für Pointer allerdings wieder intuitiv

```
typedef float* FloatPtr;   // geht, NamePtr wird aber
                           // meist anders verwendet
```

Typedef für Komplexe Typen (Vorschau)

- Bei langen, komplexen Typen kann das die Code-Lesbarkeit deutlich erhöhen

Beispiel

```
#include <vector>
typedef std::vector<float> Vec;
typedef std::vector<Vec> Mat;
int main() {
  Mat M(4,Vec(4));
  for(int i=0;i<4;++i){
    for(int j=0;j<4;++j){
      M[i][j] = (i==j);
    }
  }
  ...
}
```