Generell

`namespace`	**Katalogisierung** von Befehlen

Mit `namespace`s kann man Funktionen, Typen, etc. katalogisieren (z.B. bezüglich Projekten). Beispielsweise können Funktionen im `namespace ifee` zusammengefasst werden. Damit kann u.A. Problemen mit Namenskonflikten (z.B. unterschiedliche Funktionen mit identischem Namen) bei grösseren Projekten vorgebeugt werden.

```
namespace ifee { // namespace called ifee

    // POST: "Hi" was written to the terminal
    void output_func () { // this function is in namespace ifee
        std::cout << "Hi";
    }
}

int main () {
    ifee::output_func(); // use output_func from namespace ifee
    return 0;
}
```

Standardbibliothek

`std::min`	**Minimum** zweier Argumente

Erfordert: `#include <algorithm>`

Sonst gibt es noch:
`std::max` **Maximum** zweier Argumente

Wichtig ist, dass beide Argumente vom selben Typ sind. Sonst geht es nicht.

```
double z;
std::cin >> z;
std::cout << std::min(z, 1.0); // min of z and 1

std::cout << std::min(z, 1); // Error: z is double, 1 is int
```

Datentypen

Referenzen	**Alias** für bestehende Variable

Referenzen können nur Variablen ihrens zugrundeliegenden Typs referenzieren. Sonst gibt es einen Fehler.

Ausserdem können Referenzen nur mit l-Werten initialisiert werden (also Werten mit einer Adresse im Speicher).

```
// Usage
int a = 3;
int& b = a; // reference to a
std::cout << b << "\n"; // Output: 3
a = 18;
std::cout << b << "\n"; // Output: 18
b = 25;
std::cout << a << "\n"; // Output: 25

// Issues
int& c = 3; // Error: 3 has no address in memory
bool d = false;
int& e = d; // Error: d is bool, e wants to reference an int
```