

# Algorithmen und Datenstrukturen in C/C++

Prof. Dr. rer. nat. Peter Gerwinski

20. April 2023

# Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

## 1 Einführung

...

1.8 Templates

1.9 Container-Templates

1.10 Iteratoren

1.11 Exceptions

1.12 Typ-Konversionen

1.13 Intelligente Zeiger

## 2 Datenorganisation

## 3 Datenkodierung

## 4 Hardwarenahe Algorithmen

## 5 Optimierung

## 6 Numerik



Änderungen  
vorbehalten

# 1 Einführung in C++

## 1.13 Intelligente Zeiger

### Warum?

- bereits freigegebene Zeiger werden u. U. weiterhin verwendet
- Speicherlecks
- uninitialisierte Zeiger
  
- `shared_ptr`
- `weak_ptr`
- `unique_ptr`
- `move()`

# 1 Einführung in C++

## 1.13 Intelligente Zeiger

### Wie?

- R-Wert-Referenztypen: `&&`
- `move()`-Funktion

### Literatur:

- [http://thbecker.net/articles/rvalue\\_references/section\\_01.html](http://thbecker.net/articles/rvalue_references/section_01.html)
- <http://www.artima.com/cppsource/rvalue.html>

## 1.11 Exceptions

```
try
{
    ...
    throw <value>;
    ...
}
catch (<type> <variable>)
{
    ...
}
catch ...
```

- Nach den `catch()`-Statements wird, soweit nicht anders programmiert, das Programm fortgesetzt.
- `throw;` (ohne Wert):  
an übergeordneten Exception-Handler weiterreichen
- C-Äquivalent:  
`setjmp()`, `longjmp()`
- speziell für `<type>`:  
Nachfahren von `class exception`
- veraltet:  
*dynamic exception specifications*

## 1.12 Typ-Konversionen

- In C:

```
char *hello = "Hello, world!";  
uint64_t address = (uint64_t) hello;  
printf ("%s" PRlu64 "\n", address);
```

- alternative Syntax in C++:

```
char *hello = "Hello, world!";  
uint64_t address = uint64_t (hello);  
cout << address << endl;
```

- zusätzlich in C++:

implizite und explizite Typumwandlung zwischen Zeigern auf Klassen

```
dynamic_cast<>()  
static_cast<>()  
reinterpret_cast<>()  
const_cast<>()
```

## 1.12 Typ-Konversionen

<http://www.cplusplus.com/doc/tutorial/typecasting/>

- Zuweisung: Zeiger auf abgeleitete Klasse an Zeiger auf Basisklasse  
→ implizite Typumwandlung möglich
- Zuweisung: Zeiger auf Basisklasse an Zeiger auf abgeleitete Klasse  
→ nur explizite Typumwandlung möglich:  
`dynamic_cast<>()`, `static_cast<>()`
- implizite Typumwandlungen in der Klasse definieren:
  - Initialisierung durch Konstruktor
  - Zuweisungs-Operator
  - Typumwandlungsoperator
- implizite Typumwandlungen ausschalten:  
Schlüsselwort `explicit`

## 1.12 Typ-Konversionen

<http://www.cplusplus.com/doc/tutorial/typecasting/>

- `dynamic_cast<>()`  
Zuweisung: Zeiger auf Basisklasse an Zeiger auf abgeleitete Klasse  
explizite Typumwandlung mit Prüfung, ggf. Exception
- `static_cast<>()`  
Zuweisung: Zeiger auf Basisklasse an Zeiger auf abgeleitete Klasse  
explizite Typumwandlung ohne Prüfung
- `reinterpret_cast<>()`  
Typumwandlung ohne Prüfung zwischen Zeigern untereinander  
und zwischen Zeigern und Integer-Typen
- `const_cast<>()`  
„**const**“ ein- bzw. ausschalten