Einführung in die STL

Florian Adamsky, B. Sc. (PhD cand.)

florian.adamsky@iem.thm.de
http://florian.adamsky.it/

Softwareentwicklung im WS 2014/15

TECHNISCHE HOCHSCHULE MITTELHESSEN

Outline

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Inhaltsverzeichnis

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Typisierung

Definition (Typsystem)

schränkt den Wertebereich und die Operation von Variablen ein.

Arten

schwach	Datentyp wird nicht gespeichert → dadurch schnell
stark	Datentyp wird gespeichert
statisch	Compiler kann zur Compilezeit den Datentyp prüfen
dynamisch	Interpreter kann zur Laufzeit den Datentyp prüfen

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Beispiel Typsystem

Example (Dynamisch Javascript)

```
var number = 13;
var string = "thirteen";
var komma  = 3.14;
```

Example (Statisch C++)

```
int number   = 13;
string str   = "thirteen";
double komma = 3.14;
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Typisierung von Programmiersprachen

schwach und statisch

- C, C++
- Objective C

schwach und dynamisch

- gibt es nicht

stark und statisch

- Java

stark und dynamisch

- Python
- Perl
- JavaScript
- Ruby
- ...

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Was ist generische Programmierung?

Definition (Generische Programmierung)

Generisch bedeutet allgemeingültig oder nicht spezifisch. GP verbessert die Wiederverwendbarkeit von Code. Es befreit einen Algorithmus von festen Datentypen.

- stark und dynamisch typisierte Sprachen haben automatisch GP
- in C++ wird die GP mit *Templates* umgesetzt

Problem

Example (Klasse Util)

```
class Util {
public:
    void swap(int& a, int& b) {
        int temp = a;
        a = b;
        b = temp;
    }
};
```

Problem

Example (Klasse Util)

```
class Util {
public:
    void swap(int& a, int& b) {
        int temp = a;
        a = b;
        b = temp;
    }
};
```

Example (Main)

```
int main(void) {
    Util utils;

    int a = 10;
    int b = 23;

    cout << a << " - " << b << endl;
    utils.swap(a, b);
    cout << a << " - " << b << endl;
}
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Problem

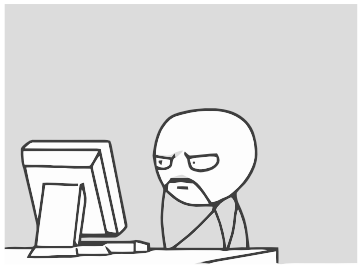
Example (Klasse Util)

```
class Uti
public:
    void
        i
        a
        b
    }
};
```

Example (Main)

```
}
```

Problem



Die Methode `tauscheVariablen()` funktioniert nur mit `int&`. Alle anderen Datentypen oder Klassen müssen erneut implementiert werden.

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Händische und umständliche Lösung

Example (Klasse Util)

```
class Util {
public:
    void swap(int& a, int& b) {
        int temp = a;
        a = b;
        b = temp;
    }

    void swap(double& a, double& b) {
        double temp = a;
        a = b;
        b = temp;
    }

    // ...
};
```

Händische und umständliche Lösung

Example (Klasse Util)

```
class Util {
public:
    void swap(int& a, int& b) {
        int temp = a;
        a = b;
        b = temp;
    }

    void swap(double& a, double& b) {
        double temp = a;
        a = b;
        b = temp;
    }

    // ...
};
```

Algorithmus ist gleich, nur die Datentypen sind unterschiedlich

Templates

- Templates erlaubt es Methoden zu schreiben für mehrere Datentypen
- Compiler erkennt mit welchen Datentypen die Methode aufgerufen wurde und generiert automatisch den entsprechenden Code
- Methodendefinition:

```
template <class identifier> method_declaration;
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Beispiel einer Template-Methode

Example (Klasse Util)

```
class Util {
public:
    template <class TYPE>
    void swap(TYPE& a, TYPE& b) {
        TYPE temp = a;
        a = b;
        b = temp;
    }
};
```

Beispiel einer Template-Methode

Example (Klasse Util)

```
class Util {
public:
    template <class TYPE>
    void swap(TYPE& a, TYPE& b) {
        TYPE temp = a;
        a = b;
        b = temp;
    }
};
```

Example (Main)

```
int main(void) {
    Util utils;

    int a = 10;
    int b = 23;
    utils.swap(a, b);
```

Beispiel einer Template-Methode

Example (Klasse Util)

```
class Util {
public:
    template <class TYPE>
    void swap(TYPE& a, TYPE& b) {
        TYPE temp = a;
        a = b;
        b = temp;
    }
};
```

Example (Main)

```
int main(void) {
    Util utils;

    int a = 10;
    int b = 23;
    utils.swap(a, b);

    double c = 10.0;
    double d = 23.0;
    utils.swap(c, d);
```

Beispiel einer Template-Methode

Example (Klasse Util)

```
class Util {
public:
    template <class TYPE>
    void swap(TYPE& a, TYPE& b) {
        TYPE temp = a;
        a = b;
        b = temp;
    }
};
```

Example (Main)

```
int main(void) {
    Util utils;

    int a = 10;
    int b = 23;
    utils.swap(a, b);

    double c = 10.0;
    double d = 23.0;
    utils.swap(c, d);

    Util obj1;
    Util obj2;
    utils.swap(obj1, obj2);
}
```

Unterschiede der Bibliotheken

STL (S)tandard (T)emplate (L)ibrary

- programmiert von Alexander Stepanov bevor es einen C++ Standard gab

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Unterschiede der Bibliotheken

STL (S)tandard (T)emplate (L)ibrary

- programmiert von Alexander Stepanov bevor es einen C++ Standard gab

Stdlib C++ (St)an(d)ard (Lib)rary

- C++ Kommitee orientierte sich stark an die STL bei der Standardisierung der Stdlib

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Unterschiede der Bibliotheken

STL (S)tandard (T)emplate (L)ibrary

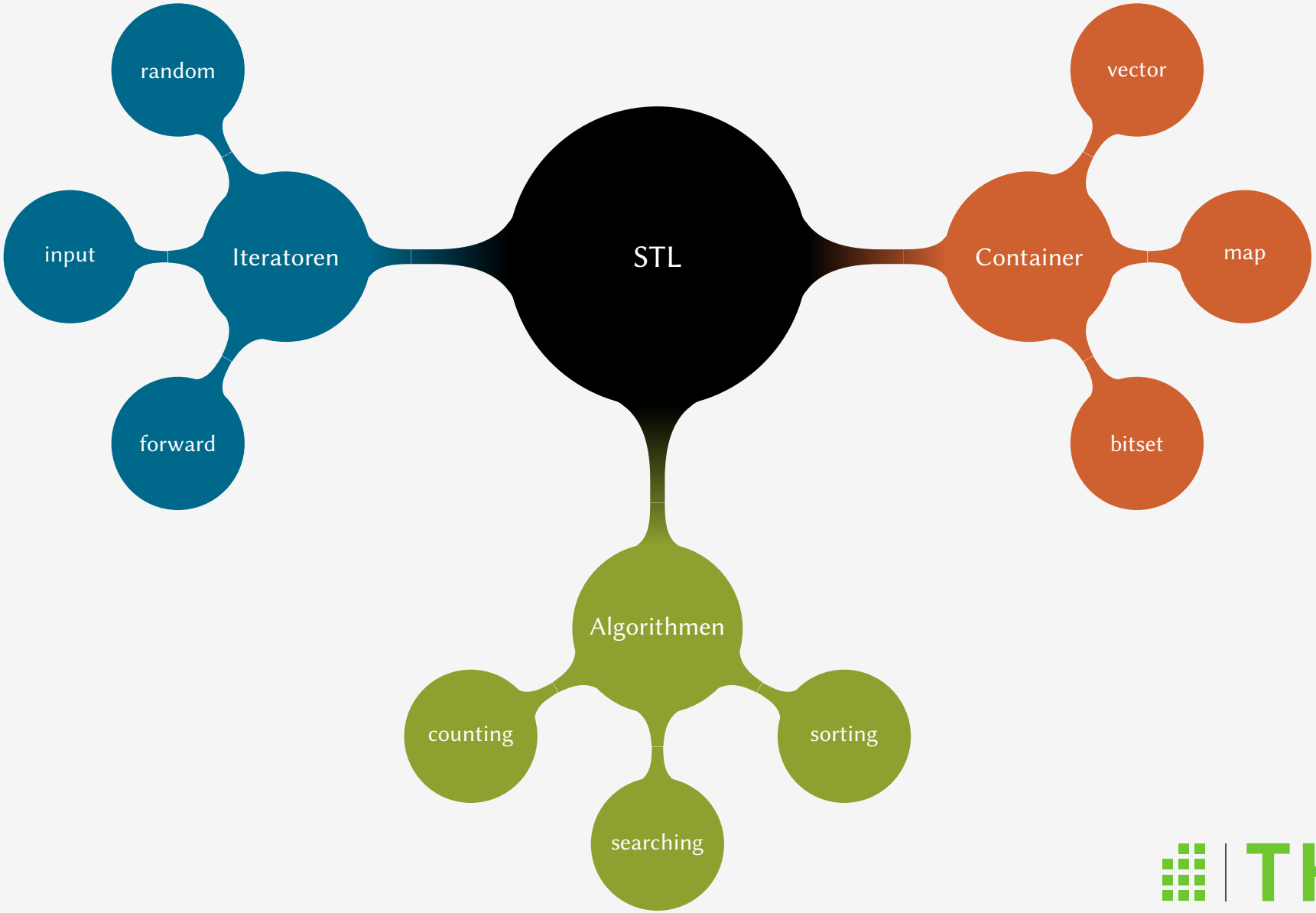
- programmiert von Alexander Stepanov bevor es einen C++ Standard gab

Stdlib C++ (St)an(d)ard (Lib)rary

- C++ Kommitee orientierte sich stark an die STL bei der Standardisierung der Stdlib

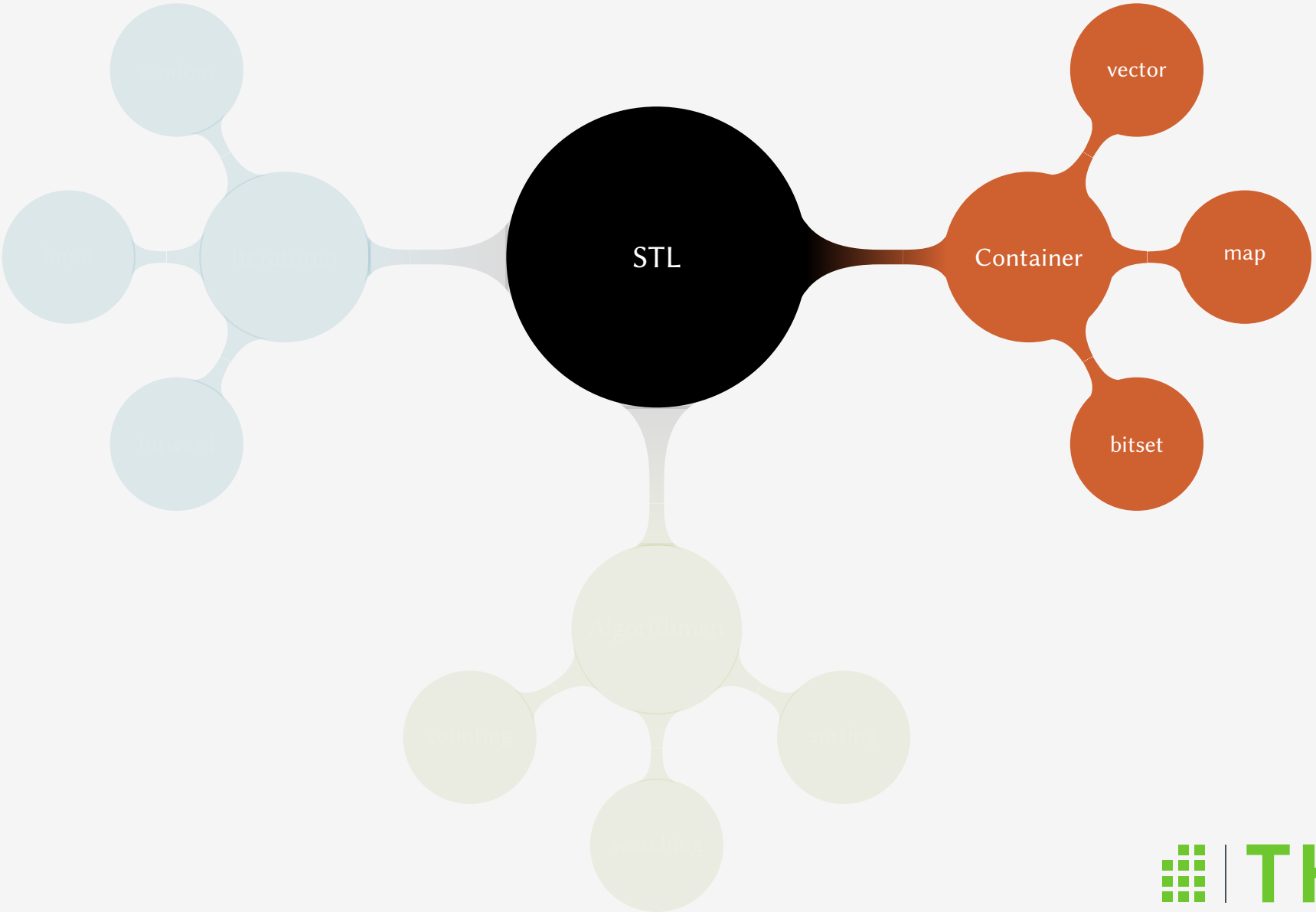
Boost Menge von freien und qualitativen Bibliotheken

Umfang der STL



THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Container



THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Vector

Definition (Vector)

ein dynamisches Array, dass während der Laufzeit neue Elemente aufnehmen kann.

```
#include <vector>
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Vector Beispiel

Example (Code)

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> vector;

    for (int i = 0; i < 100; ++i) {
        vector.push_back(i);

        cout << vector[i] << endl;
    }
}
```

Vector Beispiel

Example (Code)

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> vector;

    for (int i = 0; i < 100; ++i) {
        vector.push_back(i);

        cout << vector[i] << endl;
    }
}
```

Einbindung der `vector` Bibliothek

Vector Beispiel

Example (Code)

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> vector;

    for (int i = 0; i < 100; ++i) {
        vector.push_back(i);

        cout << vector[i] << endl;
    }
}
```

Definition eines Vectors der int-Werte speichert

Vector Beispiel

Example (Code)

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> vector;

    for (int i = 0; i < 100; ++i) {
        vector.push_back(i);

        cout << vector[i] << endl;
    }
}
```

Methode `push_back()` fügt neue Element ans Ende hinzu

Vector Beispiel

Example (Code)

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<int> vector;

    for (int i = 0; i < 100; ++i) {
        vector.push_back(i);

        cout << vector[i] << endl;
    }
}
```

Zugriff auf einzelne Element wie bei einem normalen Array

Vector Methoden

Example (Ausgangslage)

```
vector<int> vec;
```

Methode	Beschreibung
`vec.at(n)`	Liefert das n-te Element zurück
`vec.size()`	Liefert die Anzahl der Elemente zurück
`vec.push_back(val)`	Fügt `val` am Ende hinzu
`vec.pop_back()`	Entfernt das letzte Element
`vec.clear()`	Entfernt alle Element aus `vec`

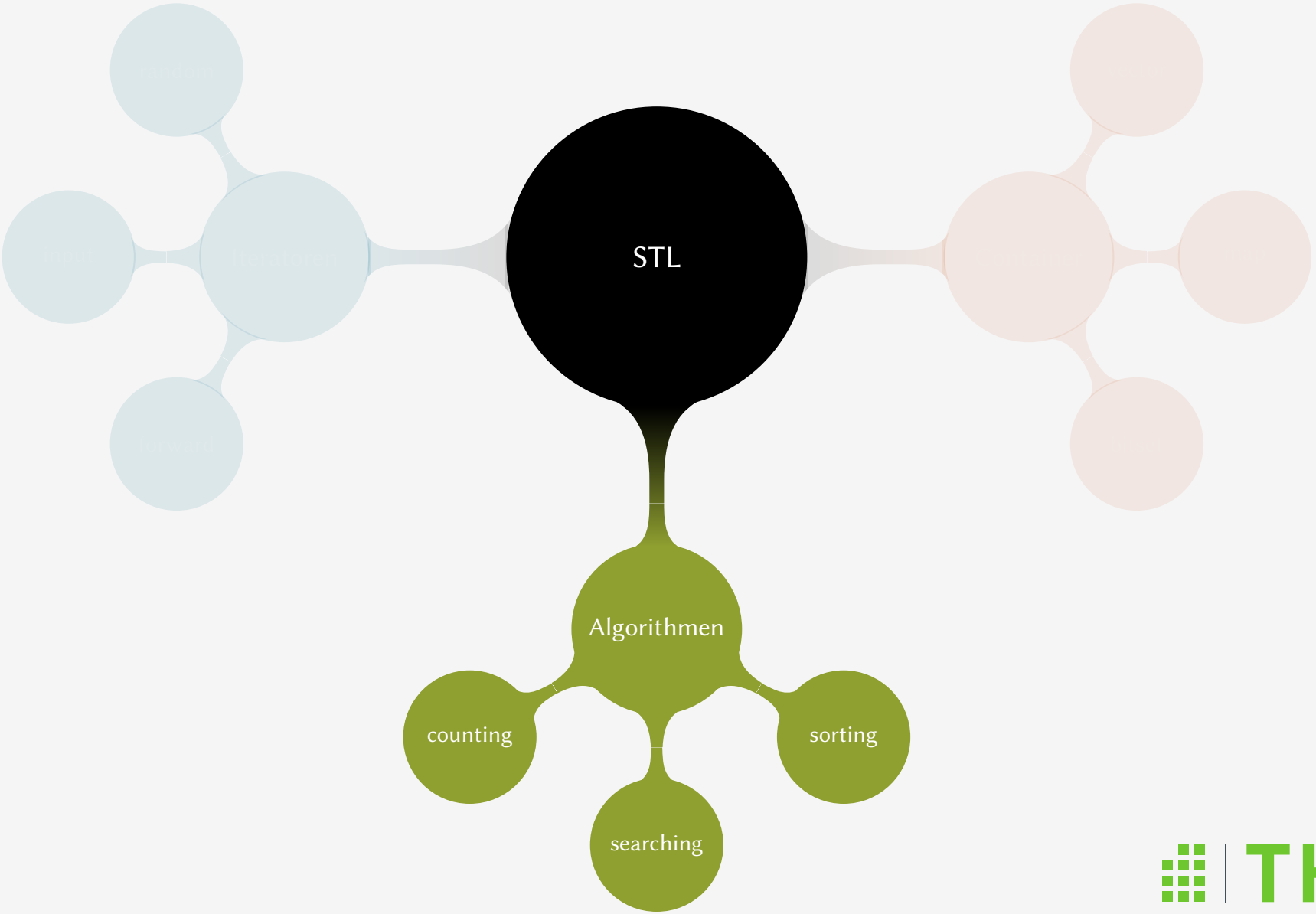
THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Weitere Container

- sequenzielle Container
  - `vector`
  - `list`
- assoziative Arrays
  - `map`
  - `set`
- andere Container
  - `bitmap`

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Algorithm



THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

algorithm

Definition (Algorithmus)

methodisches, schrittweises Verfahren das garantiert zu einer Lösung führt. Die STL-Bibliothek ist eine Ansammlung an Algorithmen aus den folgenden Problembereichen: Sortierung, Zählen, Suchen, . . .

```
#include <algorithm>
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Suchen

Example (Suchen in einem Array)

```
#include <iostream>
#include <algorithm>

using namespace std;

int main(void) {
   int  array[] = { 10, 23, 30, 42 };
   int* ptr     = find(begin(array), end(array), 23);

   if (ptr != end(array))
       cout << "Gesuchte Element: " << *ptr << endl;
}
```


THM TECHNISCHE HOCHSCHULE MITTELHESSEN

Sortieren

Example (Array sortieren)

```
#include <iostream>
#include <algorithm>

using namespace std;

bool sortierKriterium (int a, int b) {
    return (a < b);
}

int main(void) {
   int  array[] = { 32, 10, 23, 42, 100, 72, 14 };

   sort(begin(array), end(array), sortierKriterium);

   for (int i = 0; i < 7; i++)
       cout << array[i] << endl;
}
```

Zählen

Example (Elemente zählen in einem Array)

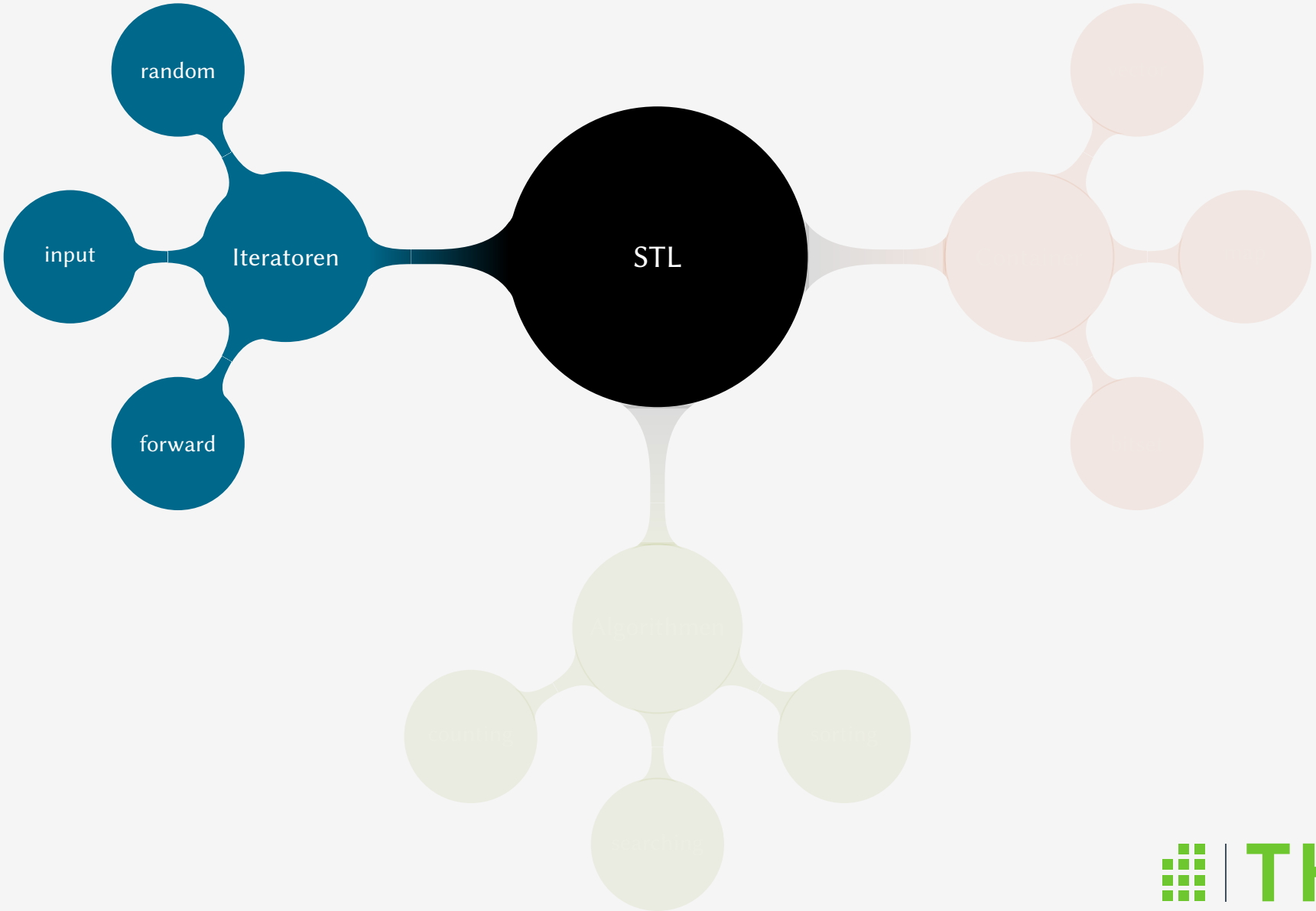
```
#include <iostream>
#include <algorithm>

using namespace std;

int main(void) {
   int array[] = { 32, 10, 23, 10, 100, 10, 14 };
   int anzahl  = count(begin(array), end(array), 10);

   cout << anzahl << endl;  // 3
}
```

Weitere Algorithmen

Funktion	Beschreibung
`for_each()`	Für jedes Element soll eine Funktion aufgerufen werden
`transform()`	Trasformiere Array mit einer Funktion
`replace()`	Ersetze den Wert x durch y
`remove()`	Entferne den Wert x
`unique()`	Entferne alle doppelten Einträge
`shuffle()`	Mische Werte in einem Array

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Iteratoren



THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Iteratoren

Definition (Iterator)

ist ein Objekt das über einen Container iteriert.

Auflistung

- Iterator ist einem Pointer sehr ähnlich
- Iteratoren trennen wiederum den Algorithmus von dem Datentyp
- verbindet die Container der STL mit den Algorithmen der STL

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Unterschied Index und Iterator

Example (Index)

```
vector<int> vec;

//...

for (int i = 0; i < vec.size(); i++) {
    cout << vec[i] << endl;
}
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Unterschied Index und Iterator

Example (Index)

```
vector<int> vec;

//...

for (int i = 0; i < vec.size(); i++) {
    cout << vec[i] << endl;
}
```

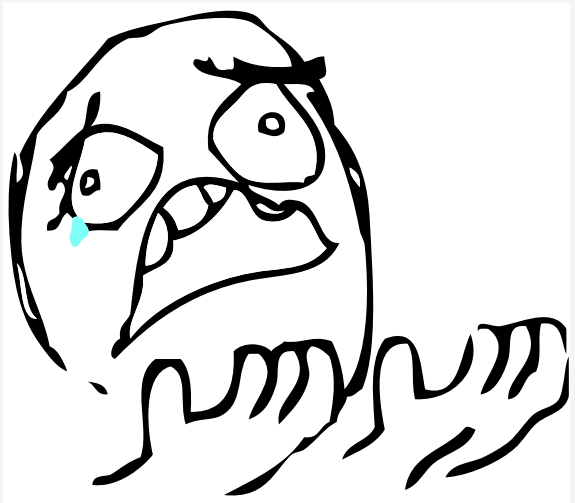
Example (Iterator)

```
vector<int> vec;

//...
vector<int>::iterator it;
for (it = vec.begin(); it != vec.end();
     ++it) {
    cout << *it << endl;
}
```


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Whyyyy?

- Index Iteration ist nicht überall möglich (z.B. `list` oder `map`)
- Flexible, da der Container einfach geändert werden kann
- die Funktionen in `<algorithm>` verlangen alle einen Iterator



Zusammenfassung

- Programmiersprachen haben unterschiedliche Typsysteme (Typisierung)
- Generische Programmierung trennt den Algorithmus von festen Datentypen
- C++ verwendet *templates* um Generische Programmierung umzusetzen
- Unterschied zwischen STL, Stdlib und Boost
- STL Container
  - `vector`
- STL Algorithmen
  - Sortierung, Zählen und Suchen
- Iterator verbindet die Container mit den Algorithmen

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN