Programmiertechniken 2

Prof. Dr.-Ing. Zhen Ru Dai
zhenru.dai@haw-hamburg.de

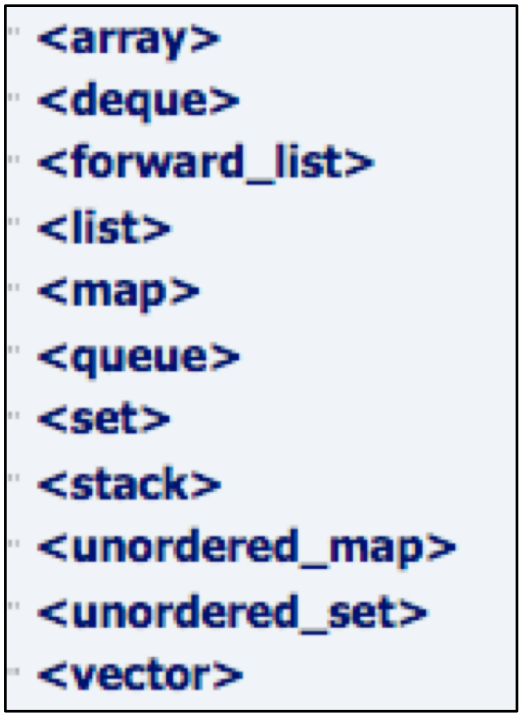
Flipped Classroom

1. Ulrich Breymann:
   - List Kapitel 27.2.3
   - Vektor Kapitel 27.2.1

2. Vectors:
   https://www.youtube.com/watch?v=Cq1h1KPoGBU&list=PL318A5EB91569E29A&index=18
   https://www.youtube.com/watch?v=iPlW5tSUOUM&list=PL318A5EB91569E29A&index=22

Standard Template Library

Die Standard Template Library (STL) ist eine Klassenbibliothek, die bei fast allen Compilern verfügbar ist.
Sie beinhaltet hauptsächlich Containerklassen für die Verwaltung von Daten, wie Listen, Stacks, Arrays usw.



Mittels Template-Parameter können Typen „**variabel**“ gehalten werden -> vgl. statische Struktur bei Array (PR 1).

Standard Template Library

T ist hier typischer Parametername

Deklaration:

```
template < class T, class Allocator = allocator<T> > class list
```

Der Typ der verwalteten Elemente muss bei der Instanziierung der Container festgelegt werden. Dazu wird in C++ der Template-Parameter verwendet. Template-Parameter werden zwischen spitzen Klammern angegeben.

Beispiel Verwendung:

```
std::list<int> first;
```

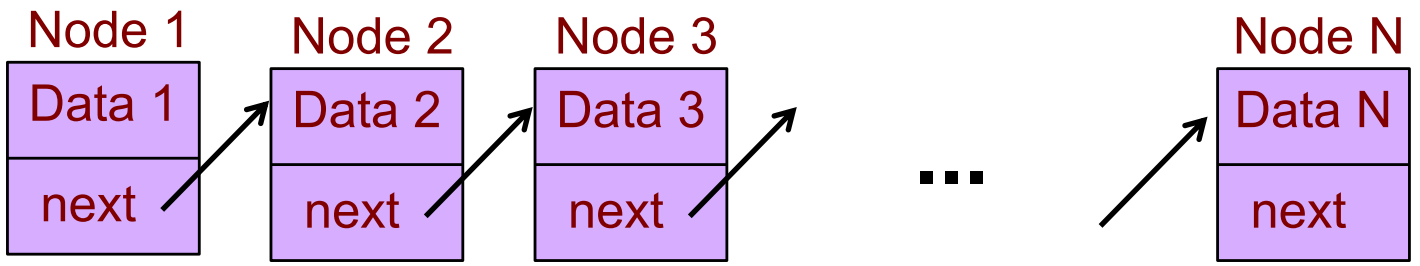
Die Standard Template Library (STL) ist im Name-Space std enthalten.

```
using namespace std;
```

Die einzelnen Containerklassen verfügen über unterschiedliche Methoden, um die Daten in die Container einzufügen oder mit ihnen zu arbeiten.

Verketterte Liste

```
struct Node{        // Datenstruktur einer Liste
     int Data;      // Element-Inhalt
     Node* next;    // Pointer auf das nächste Element
};
```



- Vorteil: Leichtes Hinzufügen und Löschen von Elementen anhand des Node-Pointers.
- Verketterte Liste in C++ Standard Template Library – std::list

Standard Template Library – std::list

Method	Result
size	Return size
front	Access first element
back	Access last element
push_front	Insert element at beginning
pop_front	Delete first element
push_back	Add element at the end
pop_back	Delete last element
insert	Insert element(s)
erase	Erase Element(s)
clear	Clear content (and calls elements destructors)

Beispiel:

```
list<Circle> cuts;
cuts.push_back(wheel1);
cuts.push_back(wheel2);
```

Typisierte Liste

Kopie an Liste anhängen

Standard Template Library – std::list

Die einzelnen Containerklassen verfügen über Überladene Operatoren. Der Index-Operator [ ] wurde überladen, sodass auf die Elemente der dynamischen Liste wie auf Elemente der statischen Liste zugegriffen werden kann

Beispiel:

```
list<Circle> cuts;
cuts.push_back(wheel1)
Circle wheel2 = cuts[0];
```

Typisierte Liste

Kopie an Liste anhängen

Index Operator

Warnung: Die Liste muss an der Index-Position ein Element besitzen. Zugriffe außerhalb der Liste führen zu Fehlern!

List-Iterator

Die Klasse List verfügt über einen eignen Iterator-Typ. Dieser Typ ist in der Klasse definiert. Der Iterator selber ist ein Pointer auf Elemente der Liste.

```
list<T>::iterator
```

Der Template-Parameter des Iterators muss dann den Typ der Elemente der Liste erhalten.

```
list<Circle>::iterator current;
```

Da die Verwaltung der Liste innerhalb der Klasse erfolgt und nach außen nicht sichtbar ist, gibt es Methoden, um auf die Enden der Liste zugreifen zu können:

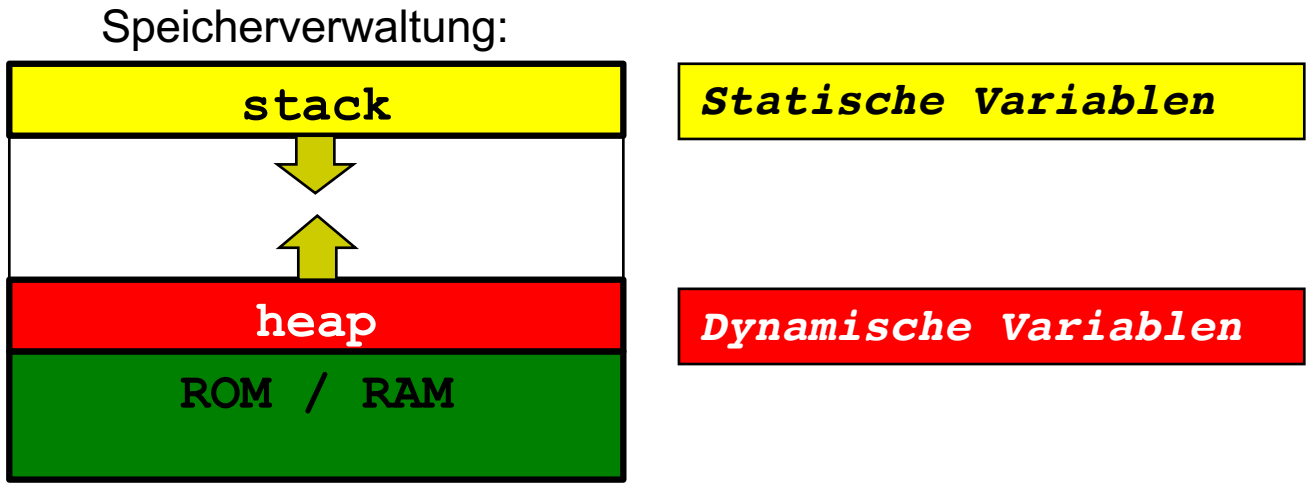
```
iterator begin();
iterator end();
```

Der Iterator-Pointer (oder einfach Iterator) kann dann mittels Increment-Operator verschoben werden.

```
list<Circle>::iterator current;
current = cuts.begin();
while (current != cuts.end()) {
  strecke += (*current).circumference();
  current++;
}
```

Standard Template Library – std::vector

- Array hat den Nachteil, dass die Array-Größe am Anfang festgelegt werden muss.
- Ein Vector ist ein Array mit dynamischer Array-Größe.
  - Vector wird auf den Heap des Speicherwerks gepackt.
  - Heap „atmet“ (d.h. wächst und schrumpft nach Bedarf).



Standard Template Library – std::vector

Method	Result
size	Return size
front	Access first element
back	Access last element
at	Access element at position
push_back	Insert element at end
pop_back	Delete last element
clear	Clear content (and calls elements destructors)
erase	Erase element or range of elements
insert	Insert elements
begin	Return iterator to beginning
end	Return iterator to end

Standard Template Library – std::vector

```
#include <vector>
#include <iostream>
using namespace std;

vector<char> v;
char c = 0;

while(c != 'x'){
   cin>>c;
   v.push_back(c);
}

vector<char>::const_iterator i;
for(i = v.begin(); i != v.end(); i++)
     cout<<*i;
```

std::list versus std::vector

- std::list

  ist eine doppelt verkettete Liste

  Daten können sehr schnell an beliebiger Stelle eingefügt und gelöscht werden

  aufwendig, einen bestimmten Datensatz per Index anzusprechen.

- std::vector

  ist ein Array mit dynamischer Länge

  man kann schnell ein bestimmtest Element lesen

  Braucht wesentlich mehr Speicher als herkömmliche Arrays