Einführung in die STL

Fimberger Lucia
lfimberg@cosy.sbg.ac.at

Nidetzky Marion
mnidetzk@cosy.sbg.ac.at

Was ist die STL?

- Abkürzung für Standard Template Library
- Eine generische Bibliothek
- Ist kaum objektorientiert, da eine Trennung von Daten und Funktionen erfolgt
- Beinhaltet die am häufigsten gebrauchten Algorithmen und Datenstrukturen
  z.B. Vektoren, Listen, Stacks, Such- und Sortieralgorithmen, ...

- Entwickelt von D. Musser, A. Saini, A. Stepanov und M. Lee

- 1994 von ANSI C++ Komitee in C++ Standard aufgenommen

Vorteile der STL

- Templates
  - Auswertung der Templates geschieht zur Kompilierzeit
  - Keine Laufzeiteinbußen
- Standardisierung
  - Leichter portierbar
  - Leichter wartbar

Organisation der STL

1. Container (Behälter)

2. Algorithmen

3. Iteratoren

Diese 3 Bereiche agieren allerdings nicht alleine, sondern sind miteinander verbunden. Einfach ausgedrückt könnte man sagen:
Algorithmen werden auf Container mittels Iteratoren angewandt.

Weitere Bereiche:

- Funktionsobjekte
- Adaptoren

Container

- Container sind Objekte, die andere Objekte enthalten
- Die STL beinhaltet verschiedene Arten von Containern
- Sind der Kern der STL
- Assoziative und sequentielle Containerklassen

Sequentielle Containerklassen

Jedes Element besitzt eine eigene Position.

Container	Beschreibung	Header
vector	ein dynamisches Array	<vector>
list	lineare Liste	<list>
deque	doppelt verkettete Reihe	<deque>

Sequentielle Containerklassen (con't)

Adaptoren für Sequenzen verleihen einer Sequenz eine andere Schnittstelle.

Container	Beschreibung	Header
stack	ein Stapel	<stack>
queue	eine Reihe	<queue>
priority_queue	ein priorisierte Reihe	<queue>

Assoziativen Container

Verwalten jeweils eine Menge von Elementen, auf die über ihren Inhalt (Wert) zugegriffen wird.

Container	Beschreibung	Header
map	speichert Schlüssel/Wert-Paare, wobei ein Schlüssel nur mit einem Wert verknüpft ist	<map>
multimap	wie map, nur dass ein Schlüssel mit einem oder mehreren Werten verknüpft sein kann	<map>
set	eine Ansammlung von Daten, in welcher ein Element nur einmal vorkommt	<set>
multiset	wie set, nur dass ein Element mehrfach vorkommen kann	<set>

Iteratoren

- Iteratoren sind Objekte, die sich wie Zeiger (Pointer) verhalten
- So wie man einen Pointer über ein Array bewegen kann, kann man sich mit einem Iterator über einen Container bewegen
- Es gibt 5 Typen von Iteratoren
- Der Iteratorentyp wird zur Kompilierzeit festgelegt, damit wird entsprechend der Containerklasse und einer auszuführenden Algorithmus der jeweils am besten geeignete Typ ausgewählt

Die Typen von Iteratoren

Hierarchisch geordnet:

Iterator	Zugriffsmöglichkeiten
Random Access	Schreiben und Lesen von Daten, sowie wahlfreier Zugriff auf Elemente
Bidirectional	Schreiben und Lesen von Daten, sowie Vorwärts- und Rückwärtsnavigation
Forward	Schreiben und Lesen von Daten, sowie Vorwärtsnavigation
Input	Schreiben von Daten, sowie Vorwärtsnavigation
Output	Lesen von Daten, sowie Vorwärtsnavigation

Algorithmen

- Arbeiten mit Iteratoren, die auf Container zugreifen
- Arbeiten auch mit *normalen* Zeigern, wegen der Allgemeinheit des Entwurfs
- Algorithmen werden verwendet um Container zu initialisieren, zu sortieren, zu durchsuchen und zu ändern
- Es existieren 60 Algorithmen, die in 8 Klassen unterteilt werden können

- Unterscheidung zwischen
  - globalen Funktionen: Der Algorithmus musste nur einmal für alle Container implementiert werden.
  - Elementfunktionen: Einige Funktionen sind auch als Elementfunktion der Klasse vorhanden.

Algorithmen (con't)

Klasse	Beschreibung	Beispiele
nichtmodifizierte Sequenzoperationen	extrahieren Information, suchen Positionen modifizieren keine Elemente	find()
modifizierte Sequenzoperationen	vielfältige Operationen, die Elemente, auf die sie jeweils zugreifen, verändern	swap(), fill()
Sortieren	sortieren und prüfen der Konsistenz (Indizes)	sort()
Mengenalgorithmen	erzeugen von sortierten Strukturen	set_union()
Heap-Operationen		make_heap()
Min und Max		min(), max()
Permutationen		next_permutation()
numerische Alg.		partial_sum()

Container: vector

- Klasse vector ist ein dynamisches Array
- Die Größe des Array kann sich während der Laufzeit ändern
- Stellt alle Methoden zur Verfügung, die man braucht um auf Elemente zuzugreifen und sie zu verändern

Container: vector (con't)

Um die STL-Klasse vector verwenden zu können, muss folgender Code eingefügt werden:

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

Nach diesen 2 Zeilen steht die vector-Klasse zur Verfügung. Da diese eine Template-Klasse ist, wird bei der Deklaration angegeben welche Art von Objekte das dynamische Array aufnimmt.
Hier einige Beispiele:

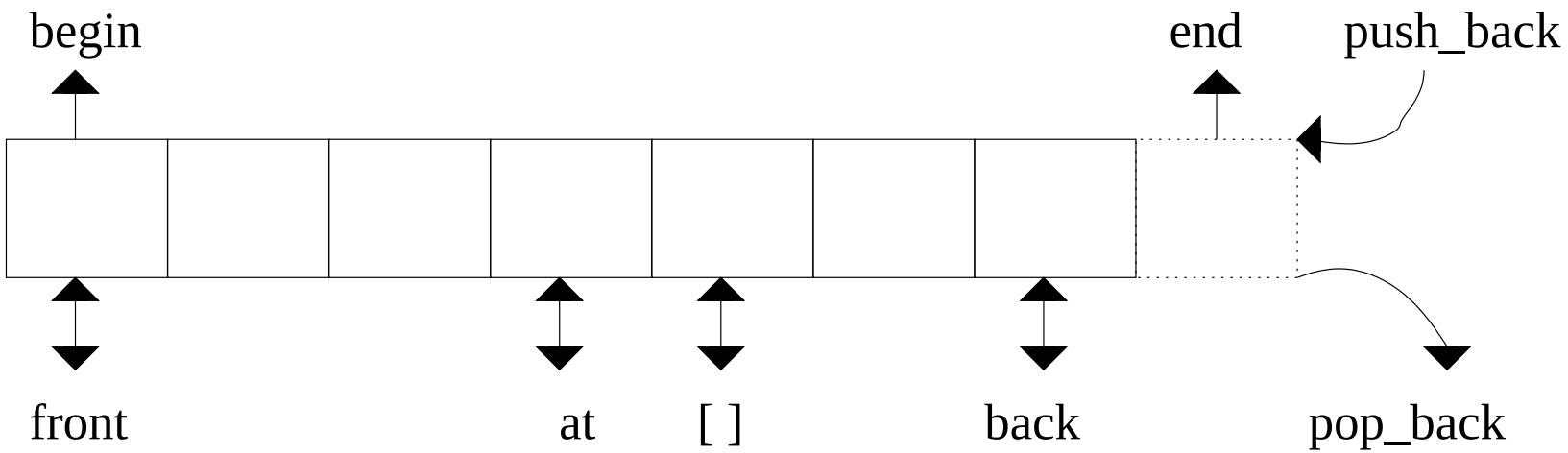
Container: vector (con't)

```
vector<int> i_vec;              // erzeugt einen integer Vector
                                // mit der Anfangslänge 0

vector<char> c_vec(20);         // erzeugt einen character Vector
                                // mit der Anfangslänge 20

vector<long> l_vec(10, 7);      // erzeugt einen long Vector mit der
                                // Länge 10, und initialisiert alle
                                // Elemente mit 7

vector<MyClass> mc_vec;         // erzeugt einen Vector, der Elemente
                                // von Typ MyClass aufnehmen kann
```

Container: vector (con't)

Methoden	Beschreibung
size()	liefert die aktuelle Größe
begin()	liefert einen Iterator auf den Anfang des vector-Objekts
end()	liefert einen Iterator auf das Ende des vector-Objekts
push_back	fügt ein Element an das Ende an
pop_back	löscht das letzte Element
insert	fügt ein oder mehrere Elemente an eine beliebige Stelle in das Objekt ein
erase()	löscht Elemente
front()	liefert Referenz auf das erste Element des Vectors
back()	liefert Referenz auf das letzte Element
at()	liefert Referenz auf ein bestimmtes Element

Container: vector (con't)



Quellen und Links

http://www.igpm.rwth-aachen.de/c++/

http://www.baeumle.com/info/cplusplus/

http://www.cs.rpi.edu/~wiseb/stl-notes.html

http://www.cplusplus.com/

http://graphics.cs.uni-sb.de/Courses/ws0102/Sopra/sw/STL-Dateien/