Informatik Übungsstunde - Woche 12

Eric Ceglie

Iteratoren

Wir wissen bereits, dass uns Pointer das Durchlaufen von zusammenhängendem Speicher erlauben. Bei Containern funktioniert dies jedoch im Allgemeinen nicht mit Pointern, da man dafür wissen müsste, wie der gegebene Container implementiert ist. Deshalb stellen die meisten Container sogenannte *Iteratoren* bereit, um sich in der gegebenen Datenstruktur zu bewegen.

Beispiele

```
#include <vector>

std::vector<int> v = {8,3,1,4,6,9};
for (std::vector<int>::iterator it = v.begin(); it != v.end(); ++it) {
        std::cout << *it << " ";
}
```

```
#include <set>

using iterator = std::set<int>::iterator;

std::set<int> s = {8,3,1,4,6,9};
for (iterator it = s.begin(); it != s.end(); ++it) {
        std::cout << *it << " ";
}
```

Wie im zweiten Besipiel veranschaulicht, bietet es sich bei Iteratoren immer an den Befehl `using` zu benutzen, um den Typen des Iterators übersichtlicher darzustellen.

Erlaube Operationen

Beachte, dass man sich Iteratoren ähnlich wie Pointer vorstellen kann, aber im Allgemeinen weniger Operationen unterstützt werden. Es ist wichtig zu wissen, dass in C++ jeder Container seine eigenen Iterator implementiert. Die folgenden vier Grundoperationen sind jedoch bei jedem Container `c` gegeben:

- `it = c.begin()` : Iterator auf das erste Element
- `it = c.end()` : Iterator auf das Ende des Containers `c`
- `*it` : Zugriff auf das aktuelle Element
- `++it` : Iterator um ein Element nach vorne verschieben