Programmierung 2 VU 051020

Übungseinheit 4

WS 2020

universität
wien

Typische Problemfälle bei Verwendung von Algorithmen

universität wien

copy, copy_if, ...

Achtung: Die Werte werden in den Zielbereich kopiert! Das heißt, es muss bereits entsprechend Platz im Zielbereich vorhanden sein.

```
vector<int> source{1,2,3,4};
vector<int> target;
copy(source.begin(), source.end(), target.begin()); //undefiniertes Verhalten!
```

Es muss zuerst Platz geschaffen werden:

```
target.reserve(source.size());
copy(source.begin(), source.end(), target.begin()); //OK?
```

universität wien

reserve

Achtung: Es wird nur entsprechend Platz reserviert, es werden aber keine Elemente eingefügt!

```
target.reserve(source.size());
copy(source.begin(), source.end(), target.begin());
//Kein undefiniertes Verhalten mehr, aber target ist noch immer leer!
```

Abhilfe:

```
target.resize(source.size());
copy(source.begin(), source.end(), target.begin()); //OK
```

universität wien

Alternativen

Da unsere Vector-Klasse nicht über die Methode `resize` verfügt und auch eine Manipulation von `sz` nicht zulässt – außer beim Einfügen von Werten – benötigen wir Alternativen. Dafür sind folgende Faustregeln definiert:

- Bevorzuge die Range Varianten der Methoden, wenn mehrere Elemente einzufügen sind.
- Bevorzuge Methoden des Containers (`insert`, `assign`) gegenüber Algorithmen (copy), da Methoden für den Container speziell auf diesen hin optimiert und damit in der Regel effizienter sind.

Aber copy_if ...

Die einzufügende Anzahl ist nicht bekannt und es gibt auch keine passende Methode in der Klasse Vector

```
target.resize(?); //Anzahl der zu kopierenden Elemente im Allgemeinen unbekannt
copy_if(source.begin(), source.end(), target.begin(),
 [] (const int& i) -> bool{return i%2;}); // Kopiere ungerade Eintraege
```

Lösung:

```
copy_if(source.begin(), source.end(), back_inserter(target),
 [] (const int& i) -> bool{return i%2;});
```

universität wien

back_inserter, front_inserter, inserter

- Spezielle Formen von Iteratoren, die, wenn Sie dereferenziert werden, ein neues Element im Container anlegen (mit `push_back`, `push_front`, `insert`) und dann auf dieses verweisen.
- `front_inserter` kann nur verwendet werden, wenn der Container `push_front` anbietet (was vector zum Beispiel nicht tut).
  Anmerkung: Die Werte werden "verkehrt herum" eingefügt. Eventuell muss man die einzufügenden Werte mit einem `reverse_iterator (rbegin(), rend())` durchlaufen.
- `inserter` kann verwendet werden, um an einer beliebigen Stelle einzufügen, beim Erstellen des Inserters muss diese Position als zusätzlicher Parameter angegeben werden (z.B. `inserter(target, target.begin()+2)`).

universität wien

Vorsicht, wenn beim Kopieren Quelle und Ziel überlappen

Die `insert` Methode liefert dann undefiniertes Verhalten. Der copy-Algorithmus funktioniert, wenn der Beginn des Ziels außerhalb der Quelle liegt, der copy_backward-Algorithmus muss verwendet werden, wenn das Ende außerhalb der Quelle liegt.

Anmerkung: Anfang und Ende des Ziels können nicht beide innerhalb des Quellbereichs liegen (außer, wenn beide Bereiche übereinstimmen, was aber eine Kopie sinnlos macht). Warum?

remove, remove_if ...

Achtung: remove löscht nicht!

```
vector<int> source{1,2,3,4};
remove(source.begin(),source.end(),2);
cout << source.size(); //4! Inhalt {1,3,4,?}
```

Remove verschiebt nur alle nicht zu löschenden Elemente (unter Beibehaltung der Reihenfolge) an den Beginn des Containers und retourniert einen Iterator auf das erste "ungültige" Element. Der Inhalt ab diesem Iterator ist nicht mehr spezifiziert (können Kopien der ursprünglich dort vorhandenen Elemente sein, aber auch ganz etwas anderes). Die unnötigen Einträge können aus dem Container mittels erase entfernt werden $\rightarrow$ erase-remove-Idiom.

```
source.erase(remove(source.begin(),source.end(),2),source.end());
cout << source.size(); //3 OK Inhalt {1,3,4}
```

universität wien

Aufgabe

Der Vektor v enthalte eine gerade Anzahl von Elementen. Vektor $v1$ ist vom gleichen Typ und nicht leer. Es soll $v1$ so verändert werden, dass sein Inhalt genau der zweiten Hälfte des Inhalts von v entspricht Beispiel: v enthält $\{1, 2, 3, 4\}$, $v1$ $\{7, 8, 9, 10\}$. Nach der Aktion soll $v1$ $\{3, 4\}$ enthalten.

Ideen?

universität wien

Lösung

```
v1.assign(v.begin()+v.size()/2, v.end());
```

Auch OK:

```
v1.clear();
v1.insert(v1.end(), v.begin()+v.size()/2, v.end());
```

bzw.

```
v1.clear();
copy(v.begin()+v.size()/2, v.end(), back_inserter(
    v1)); //Effizienz?
```

Nur diese Lösung funktioniert auch mit unserer Vector-Klasse ohne Erweiterungen.
Wenn Sie sich eine eigene Schleife ausgedacht haben: Shame on you!

universität wien

STL Übungsaufgaben

Lösen Sie die Übungsaufgaben zu STL (siehe separates PDF-Dokument in Moodle).

Ziele für nächste Woche

In der nächsten Übungsstunde können Sie Ihren Vector in der Prüfungsumgebung Moped im Rahmen eines Probetests prüfen.

Reminder:
Falls Sie am Probetest teilnehmen möchten, müssen Sie Ihren Vector bis 03.11.2020 auf Moodle hochladen.

Die Abgabefrist für Ihren Vector für den Test ist am 10.11.2020.

universität wien