Programmierung 2 VU 051020

Übungseinheit 2

WS 2020

universität
wien

Vector – Überblick

Auf Basis der Klasse Vector wie zu Folien für Woche 1 beschrieben ist Folgendes bis zur nächsten Woche zu erledigen

Iteratoren **Iterator** und **ConstIterator**. Ermöglichen es range-based-for-loops zu verwenden sowie Algorithmen der STL.

Testen Testen Sie ihre Implementierung mit den verfügbaren Testprogrammen (siehe Lernplattform).

universität wien

Iteratoren – Wiederholung

- Iteratoren sind eine Verallgemeinerung des Pointerkonzepts (oder des Referenzkonzepts). Ein Iterator referenziert einen Wert in einem Container. Es werden zumindest die beiden Operatoren * (Dereferenzieren) und ++ (Prefix Inkrement; Weiterschalten zum nächsten Wert) angeboten.
- Weitere Operationen sind optional. Im Kontext mit range-based-for-loops ist != (Vergleich zweier Iteratoren) notwendig (meist wird dann auch gleich == angeboten).
- Der C++-Standard definiert einige Iterator-Typen, die unterschiedliche Operationen anbieten. Es ist genau festgelegt, welche Operationen ein bestimmter Typ anbieten muss. Die Details sind komplex.
- Unsere Iteratoren sind Forward-Iteratoren.

Kompatibilität unserer Iteratoren mit STL-Algorithmen

Damit Iteratoren von STL-Algorithmen verwendet werden können, müssen einige Typ-Aliase für die Iteratoren angelegt werden.

Am einfachsten definiert man diese am Anfang der Vector-Klasse:

```
class Vector{
public:
  class ConstIterator;
  class Iterator;
  using value_type = double;
  using size_type = std::size_t;
  using difference_type = std::ptrdiff_t;
  using reference = value_type&;
  using const_reference = const value_type&;
  using pointer = value_type*;
  using const_pointer = const value_type*;
  using iterator = Vector::Iterator;
  using const_iterator = Vector::ConstIterator;
private:
  //Instanzvariablen
public:
  //Methoden
};
```

- Sorgen Sie dafür, dass Ihre Vector-Klasse nur noch die Datentypen aus den using-Deklarationen verwendet.

universität wien

Vorsicht: const_iterator $\neq$ const iterator

Ein const_iterator ist kein konstanter Iterator!

Vielmehr liefert ein const_iterator eine konstante Referenz auf den referenzierten Wert. Das heißt, die Dereferenzoperation (operator$*$) erlaubt nur lesenden Zugriff auf das im Vector gespeicherte Element.

const_iterator ist also eine eigenständige Klasse mit einer eigenen Implementierung.

universität wien

Iterator

```
class Vector {
  (...)
  class Iterator {
    public:
      using value_type = Vector::value_type;
      using reference = Vector::reference;
      using pointer = Vector::pointer;
      using difference_type = Vector::difference_type;
      using iterator_category = std::forward_iterator_tag;
    private:
    //Instanzvariablen
    public:
    //Methoden
  };
  class ConstIterator {
    public:
      using value_type = Vector::value_type;
      using reference = Vector::const_reference;
      using pointer = Vector::const_pointer;
      using difference_type = Vector::difference_type;
      using iterator_category = std::forward_iterator_tag;
    private:
    //Instanzvariablen
    public:
    //Methoden
  };
};
```

universität wien

Iteratoren für unsere Vector-Klasse

Erweitern der Klasse Vector um die Methode begin() und end().

iterator begin() Liefert einen Iterator auf das erste Element im Vektor. Ist der Vektor leer, entspricht er dem end-Iterator

iterator end() Liefert einen Iterator auf das element nach dem letzen Element im Vektor.

const_iterator begin() const Liefert einen Iterator auf das erste Element im Vektor. Ist der Vektor leer, entspricht er dem end-Iterator

const_iterator end() const Liefert einen Iterator auf das element nach dem letzen Element im Vektor.

universität wien

Iterator – Implementierung

Instanzvariablen

- `pointer ptr` Zeiger auf ein Element im Vector.

Konstruktoren

- `Default` Liefert einen Iterator auf **nullptr**.
- `pointer ptr` Liefert einen Iterator, der die Instanzvariable auf ptr setzt.

Methoden

`reference operator*()` const?

`pointer operator->()` const?

`bool operator==(const const_iterator&)` const?

`bool operator!=(const const_iterator&)` const?

`iterator& operator++()` const? (Prefix)

`iterator operator++(int)` const? (Postfix)

`operator const_iterator()` const? (Typ-Konversion)

universität wien

ConstIterator – Implementierung

Instanzvariablen

pointer ptr Zeiger auf ein Element im Vector.

Konstruktoren

Default Liefert einen ConstIterator auf **nullptr**.

pointer ptr Liefert einen ConstIterator, der die Instanzvariable auf ptr setzt.

Methoden

reference operator*() const?

pointer operator->() const?

bool operator==(const const_iterator&) const?

bool operator!=(const const_iterator&) const?

const_iterator& operator++() const? (Prefix)

const_iterator operator++(int) const? (Postfix)

universität wien

Iteratoren – Anmerkungen

- Erweitern Sie die Vector Klasse um die Methoden **iterator begin()**, **iterator end()**, sowie die dazugehörigen const methoden für const_iterator.
- Überlegen Sie sich, welche der (Const)Iterator-Methoden **const** sein müssen und welche nicht.
- Wofür wird der const_iterator Typ-Konversions-Operator in der Iterator-Klasse benötigt?
- Warum wird für die Vergleichsoperatoren == bzw. != der Iterator-Klasse **const_iterator** als Parameter verwendet?

universität wien

Vector – Insert/Erase

Die Methoden `insert` und `erase` können von den VO-Folien kopiert werden.

```
iterator insert(const_iterator pos, const_reference val) {
  auto diff = pos-begin();
  if (diff<0 || static_cast<size_type>(diff)>sz)
    throw std::runtime_error("Iterator out of bounds");
  size_type current{static_cast<size_type>(diff)};
  if (sz>=max_sz)
    reserve(max_sz*2);
  for (auto i {sz}; i-->current;)
    values[i+1]=values[i];
  values[current]=val;
  ++sz;
  return iterator{values+current};
}
```

universität wien

Vector – Insert/Erase

Die Methoden `insert` und `erase` können von den VO-Folien kopiert werden.

```
iterator erase(const_iterator pos) {
  auto diff = pos-begin();
  if (diff<0 || static_cast<size_type>(diff)>=sz)
    throw std::runtime_error("Iterator out of bounds");
  size_type current{static_cast<size_type>(diff)};
  for (auto i{current}; i<sz-1; ++i)
    values[i]=values[i+1];
  --sz;
  return iterator{values+current};
}
```

Iterator/ConstIterator – Operator-

Damit Methoden `insert` und `erase` funktionieren muss

```
friend Vector::difference_type operator−(const Vector::ConstIterator& lop,
      const Vector::ConstIterator& rop) {
  return lop.ptr−rop.ptr;
}
```

vorhanden sein.

universität wien

Vector – Testfiles

Zum Testen Ihrer Implementierung sollten Sie ein eigenes main-File schreiben.
Als Ergänzung werden jede Woche Testfiles bereitgestellt.
Beachten Sie, dass diese nur als Hilfestellung gedacht sind und **keine Garantie** darstellen, dass Ihr Vector korrekt funktioniert.

Beschreibung Testfiles – Vector & Iteratoren (1)

Unter **/home/Xchange/PR2/ue2** und auf Moodle finden Sie die folgenden Testdateien. Versuchen Sie die Dateien mit Ihrer Vector-Klasse zu kompilieren und zu binden.

Wenn **types_test.cpp** nicht kompiliert, haben Sie zumindest eines der in unserem Kontext notwendigen Typ-Aliase nicht (ordnungsgemäß) definiert.

Wenn **const_test.cpp** kompiliert, haben Sie const_iterator nicht ordnungsgemäß implementiert oder (noch schlimmer) gar keine eigene ConstIteratorklasse geschrieben. Lassen Sie das Programm laufen und überlegen Sie, warum das Ergebnis so eigentlich nicht sein darf.

Beschreibung Testfiles – Vector & Iteratoren (2)

Wenn **const_test_arrow.cpp** kompiliert, ist Ihre Implementierung des Operators −>fehlerhaft.

utest_2.cpp sollte ohne Fehler kompilieren.

universität wien

Vector – Erweiterungen

Sie können zu Übungszwecken noch weitere Methoden und globale Funktionen implementieren (siehe Vorlesungsfolien PR1).
Beachten Sie ebenso die Übungsaufgaben auf den VO-Folien.
Folgende (verpflichtende) Erweiterung wird in der nächsten Übungsstunde vorgestellt:

- Templates, ermöglichen es "beliebige" Werte zu speichern, nicht nur double-Werte. (Woche 3)

Machen Sie sich bereits vor den Übungsstunden mit den Konzepten vertraut (bereitgestellte Videos).
Zum Test benötigen Sie einen Vector mit der beschriebenen Basisfunktionalität, Iteratoren und Templates.

universität wien