Programmierung 2 VU 051020

Übungseinheit 3

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

Templates Ermöglichen es "beliebige" Werte zu speichern, nicht nur double-Werte.

Testen Testen Sie ihre Implementierung mit den verfügbaren Testprogrammen (Moodle oder almighty).

universität wien

Templates – Wiederholung (Details siehe VO-Folien)

- Templates bieten die Möglichkeit, die Typen von Variablen und Funktionsparametern zu parametrisieren (parametrischer Polymorphismus).
- Für bestimmte Werte (Typen) von Typparametern können spezielle Versionen der Templatefunktion definiert werden ("Template-Spezialisierung").
- Wird eine Template Klasse verwendet, so sind alle Methoden der Klasse implizit Templatefunktionen.

Vorgehensweise bei der Implementierung

1. Klasse Vektor wird zu einem Template mit einem Typparameter

```
template<typename T>
class Vector {...};
```

2. Ersetzen des Datentyps double als Elementtyp durch T (weniger Arbeit, wenn man schon fein säuberlich die Typ-Aliase verwendet hat, ansonsten jetzt eine gute Möglichkeit, das nachzuholen).
3. Die Definitionen der Template-Methoden kommen ebenfalls in die Header-Datei (vector.h). Diese Definitionen werden vom Compiler für die Instanzierung bei Bedarf benötigt.
4. Eventuelle Fehler beheben und Testen mit unterschiedlichen Datentypen.

universität wien

Beschreibung Testfiles – Vector & Iteratoren & Templates

Unter **/home/Xchange/PR2/ue3** und Moodle finden Sie die Datei **utest_3.cpp**. Bei der Ausführung sollten keine Fehler auftreten.

Des Weiteren finden Sie unter **/home/Xchange/PR2/ue3** die Dateien **template_test.cpp**, **miniatur.h** und **miniatur.o**.

Versuchen Sie diese Dateien mit Ihrer Vektorklasse zu kompilieren.[1]

[1] Kompilieren Sie diese am almighty.

universität wien

Aufgaben

Lösen Sie zur Testvorbereitung folgende Aufgaben.

universität wien

Aufgaben

Beantworten Sie die Fragen, die als Kommentare am Ende des Programms stehen bzw. führen Sie die erforderlichen Änderungen im Programm durch.

universität wien

Spezialisierung von `push_back(x)` für `Vector<Spezial_Miniatur>`

Spezialisieren Sie die Methode `push_back(x)` fuer Vektoren von Spezialminiaturen (`Vector<Spezial_Miniatur>`) so, dass der Parameter `x` zweimal hinten angefuegt wird.

universität wien

Speicherung von Pointern auf Miniaturen in Vector

Erstellen Sie einen `Vector<Miniaturen*>`, mit Pointern, die auf die in `vsm` gespeicherten Spezial_Miniaturen zeigen. Verwenden Sie diesen Vektor, um zu prüfen, dass mit Pointern das slicing-Problem nicht auftritt (also die korrekten virtuellen Methoden aufgerufen werden).

Ziele für nächste Woche

Sie können zu Übungszwecken noch weitere Methoden und globale Funktionen implementieren (siehe VO-Folien PR1). Beachten Sie ebenso die Übungsaufgaben auf den VO-Folien.

In der nächsten Übungsstunde werden Übungsaufgaben zu STL Algorithmen bereitgestellt.

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

Wichtige Hinweise zur Vorbereitung auf den ersten Test

Ihre Klasse Vektor darf nur die für die Implementierung notwendigen Dinge enthalten. Insbesondere nicht Lösungen von Prüfungsbeispielen aus dem Vorsemester, aus vorhergehenden Gruppen oder Ähnliches!

Vektorklassen, die nicht erlaubte Teile enthalten, werden als **Versuch der Erschleichung einer Prüfungsleistung** geahndet (ohne weitere Vorwarnung).

Abgabefrist: 10.11.2020

Falls Sie am Probetest am 04.11.2020 teilnehmen möchten, müssen Sie Ihren Vektor bereits bis 03.11.2020 hochladen. (Nochmaliges Hochladen bis 10.11.2020 ist möglich.)

universität wien

Wichtige Hinweise zum Test und zur Bewertung

- Die Klassen, welche von der LV-Leitung gegeben werden, dürfen nicht verändert oder erweitert werden.
- Die Konsistenz eines Objekts darf nicht verletzt werden (get und set-Methoden müssen sicher sein).
- Global zugängliche Variablen sind zur Lösung einer Teilaufgabe nicht erlaubt.
- Schleifen und `for_each` sind ebenfalls nicht zum Lösen mancher Teilaufgaben erlaubt. Schleifen in der Mindestvoraussetzung sind erlaubt, z.B. in `operator<<`.

universität wien