Programmieren in C++

SS 2010

Vorlesung 9, Mittwoch 23. Juni 2010

(STL, vector, string, sort, namespaces)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Erfahrungen mit dem 8. Übungsblatt
  - Nächste Woche bin ich nicht da ... aber Jens
  - Ein paar Vorab-Infos über das Projekt am Ende
- Die STL = Standard Template Library
  - Klassen für Sachen, die man oft braucht
  - zum Beispiel std::vector = ähnlich zu unserer Array Klasse
  - zum Beispiel std::string = ähnlich zu unserer String Klasse
  - zum Beispiel std::sort zum Sortieren von beliebigen Objekten
  - und was heißt dieses std::... ?

  - Übungsblatt: Schreiben Sie einen anagram finder unter Verwendung von std::vector, std::string und std::sort

Erfahrungen mit dem 8. Übungsblatt

UNI FREIBURG

- Zusammenfassung / Auszüge
  - Schöne Aufgabe
  - Wieder leichter als das letzte Mal
  - Templates wurden gut erklärt
  - Gut dass man alte Sachen wiederverwenden konnte
  - Gestaltungsfreiheit war schön ...
  - ... insbesondere auch mit Bezug auf den Schwierigkeitsgrad
  - Bitte auf diesem Niveau bleiben
  - Das Makefile war unvollständig / nicht korrekt / etc.

Nächste Woche bin ich nicht da

- Dafür Jens Hoffmann
  - der dann die Vorlesung halten wird
  - ansonsten alles ganz normal wie immer
  - insbesondere gibt es (hoffentlich) eine Aufzeichnung
  - das Übungsblatt für nächste Woche werde ich vorher mit Jens zusammen machen
  - Ich bin vom 26. Juni – 5. Juli weg (Konferenzen)
    - in der Zeit werde ich höchtens mal sporadisch auf dem Forum antworten
    - ich hoffe Jens et al übernehmen das

Das Projekt am Ende der Vorlesung 1/2

UNI FREIBURG

- Ein paar Vorab-Infos dazu
  - wir fangen ca. zwei Wochen vor Ende der Vorlesungszeit damit an
  - in diesen beiden Wochen gibt es dann keine Übungsblätter mehr und in den Vorlesungen machen wir nichts entscheidend Neues mehr, nur die eine oder andere nützliche Sache (z.B. debugging mit gdb)
  - danach noch ca. drei Wochen, also insgesamt Umfang von ca. 5 Übungsblättern

    (das Projekt macht ja auch 1/3 der Abschlussnote aus)
  - Wenn Sie erstmal weg sind nach Vorlesungsende, können Sie es auch (etwas) später noch machen
  - Haben Sie sich schon angemeldet?

Das Projekt am Ende der Vorlesung 2/2

UNI FREIBURG

- Art des Projektes
  - Ein, zwei Aufgaben zur Auswahl, von verschiedenem Schwierigkeitsgrad + evtl. die Möglichkeit ein eigenes Projekt zu definieren
  - Sie schreiben dann ein C++ Programm nach der Art, wie wir es in den Übungsblättern gemacht haben, mit Klassen, Tests, und allem
  - Nur halt etwas größer
  - Nach zwei Wochen sollten Sie einen Entwurf der .h Dateien und des Main Programmes haben und Ihrem Tutor schicken
  - In der restlichen Zeit können Sie dann alles implementieren

Die STL

- STL = Standard Template Library
  - Voller nützlicher Klassen, die man immer wieder braucht
    - vector (unser Array), string (unser String), etc.
  - Dank templates können alle diese Klassen für alle möglichen Arten von Objekten benutzt werden können
    - vector<int>, string<w_char>, ...
  - Alle Klassen in der STL stehen im std namespace
    - deswegen muss man schreiben std::vector<int> etc.
    - Erklärung dazu nächste Folie

Namespaces

- Wer, wie, was, warum
  - Gruppen von Klassen die zu einem bestimmten Projekt oder zu einer Bibilothek gehören macht man oft in einen eigenen sogenannten namespace
  - konkret heißt das, dass einfach überall darum steht

```
namespace std
{
  ...
}
```

  - Wenn man etwas aus diesem namespace benutzt, muss man dann den Namen des namespace gefolgt von :: davor schreiben, also z.B. std::vector<int>
  - Grund: Ich will ja vielleicht meine eigene Klasse vector schreiben (und sie genauso nennen)

std::vector

- Wer, wie, was, warum
  - Ein dynamisches Feld von Objekten, ähnlich zu unserem Array

```
std::vector<int> numbers;
numbers.push_back(1);
numbers.push_back(2);
for (size_t i = 0; i < numbers.size(); i++)
  printf(%d\n, numbers[i]);
```

  - size() liefert eine Variable vom Typ size_t, die je nach System 32 oder 64 bit hat, und keine negativen Werte annehmen kann. Deswegen an solchen Stellen Zählervariablen als size_t deklarieren und nicht als int
  - Siehe Codebeispiel in vorlesung-9 und Links am Ende

std::string

- Wer, wie, was, warum
  - Komfortable Klasse für Zeichenketten, ähnlich zu unserer Klasse String vom 6. Übungsblatt

    ```
    std::string s = "Hullo";
    size_t pos = s.find('u');
    if (pos != std::string::npos) s[pos] = 'a';
    printf("%s\n", s.c_str());
    ```

  - Wie beim std::vector ist die size() vom Typ size_t und ebenso Positionen in einem std::string
  - Intern benutzt die Klasse einen null-terminierte C-Strings, und den bekommt man mit der c_str() Methode

    Das ist wichtig für die printf Methode
  - Siehe Codebeispiel in vorlesung-9 und Links am Ende

std::sort

- Wer, wie, was, warum
  - Sehr oft will man Objekte sortieren, und das geht einfach und effizient mit std::sort

```
#include <algorithm>
#include <vector>

std::vector<int> numbers;
numbers.push_back(2);
numbers.push_back(1);
numbers.push_back(3);
std::sort(numbers.begin(), numbers.end());
```

  - Benutzt den operator< für Vergleiche, in diesem Fall hier den eingebauten Vergleich für int
  - Siehe Codebeispiel in vorlesung-9 und Links am Ende

Literatur / Links

- Standard Template Library (STL)
  - http://www.sgi.com/tech/stl
- std::vector
  - http://www.sgi.com/tech/stl/Vector.html
- std::string
  - http://www.sgi.com/tech/stl/basic_string.html
- std::sort
  - http://www.sgi.com/tech/stl/sort.html
- Namespaces
  - http://www.cplusplus.com/doc/tutorial/namespaces