Programmieren in C++

SS 2012

Vorlesung 7, Dienstag 19. Juni 2012

(Eingabe/Ausgabe, Optionen, ASSERT_DEATH)

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 6. Übungsblatt
- Themen heute
  - Ein- und Ausgabe
  - Kommandozeilenoptionen
  - ASSERT_DEATH
  - Übungsblatt: ein grep-ähnliches Programm
    - Gegeben eine Datei und ein "Muster", finde alle Zeilen die zu dem Muster passen
    - Dabei können Sie Ihre String Klasse wiederverwenden
    - Ich mache die wichtigsten Sachen vor

3, 5, 7, 11, ...

Erfahrungen mit dem Ü6 (String Klasse)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 19.6 13:19
  - "Test-driven development" fanden die meisten gut
    - so rum seien die Tests auch viel sinnvoller
  - Viel Zeit ging für (blöde) Speicher Fehler drauf
  - StringTest.cpp hat nicht 100% geklärt was zu tun ist
  - findAndReplace ging einfach mit find, erase und insert
  - Inkonsistenz Vorlesungscode / Folien bez. 0 am Ende
  - Fehler in der insert Methode bez. Einfügen am Ende
  - Pause ist gut, Pause ist doof … BSB nicht verpassen
  - Die Operatoren & * -> sind noch nicht ganz klar
  - Variablenzuweisung in Schleifenbedingung … gar nicht
  - Die 0 am Ende der Strings ist nicht das ASCII Zeichen 0

Erfahrungen mit dem Ü6 (String Klasse)

UNI FREIBURG

- Fortsetzung …
  - Alles auf einmal geschrieben, dann unendlich viele Fehler
  - Übersicht über die wichtigsten Sachen auf einem Blatt
  - Interner Fehler mit Jenkins (bei ein paar) … wir rätseln noch
  - Wenn ich frage ob es noch Fragen gibt, traut man sich eher noch weniger
  - Gefühlte 20% sind nicht mehr voll dabei

Ein- und Ausgabe 1/4

UNI FREIBURG

- Wir machen das erstmal C-style
  - Das heißt mit FILE*, fopen, fclose, fgets, fprintf, usw.
  - In C++ gibt es dafür sogenannte streams und die Operatoren << und >>
  - Die basieren aber auf templates, und die kommen erst nächste Woche dran (aus gutem Grund)
  - Aus unter anderem dem Grund benutzen wir auch noch printf (und heute fprintf) anstatt cout
  - Außerdem sind FILE*, fopen, etc. maschinennäher (gut für's Verständnis) und effizienter
  - Insbesondere basieren die C++ streams auf FILE*
  - Und einfacher in der Benutzung sind C++ streams auch nicht

Ein- und Ausgabe 2/4

UNI FREIBURG

- Unterschied Datei / Bildschirm?
  - Das ist in der Unix/Linux–Welt im Prinzip dasselbe
    - Auch die Benutzer Ein- und ausgabe sind Dateien
    - Auf der Kernel-Ebene heißen die alle Dateideskriptoren
      - das sind einfach ganze Zahlen, die als ID dienen
  - Die Benutzereingabe heißt standard input oder abgekürzt stdin, die Benutzerausgabe heißt standard output oder abgekürzt stdout
  - Für die Ausgabe von Fehlern gibt es noch standard error oder abgekürzt stderr; die geht standardmäßig auf den Bildschirm so wie stdout, kann aber umgeleitet werden

Ein- und Ausgabe 3/4

UNI FREIBURG

- Die wichtigsten Befehle im Überblick
  - fopen öffnet eine Datei, liefert FILE* zurück
  - fclose schließt die Datei wieder
  - feof sagt, ob wir schon am Ende der Datei sind
  - fread liest eine gegeben Anzahl Bytes aus einer Datei
  - fwrite schreibt eine gegebene Anzahl Bytes in eine Datei
  - fprintf schreibt formatiert in eine Datei, analog zu printf
  - fgets liest die nächste Zeile aus einer Datei
  - Code dazu schreiben wir gleich zusammen
  - Ansonsten alle Details auf den Linux **man pages**
    - man fopen, man fprintf, etc.

Ein- und Ausgabe 4/4

UNI FREIBURG

- Ein paar Besonderheiten
  - Wenn das Öffnen mit fopen fehlschlägt, wird NULL zurückgegeben
  - Unbedingt testen, sonst gibt es einen segmentation fault beim nächsten fgets, fwrite, etc.
  - Das Ende der Datei ist wie ein eigenes Zeichen
    - EOF = end of file
    - Insbesondere heißt das, dass wenn man das letzte richtige Zeichen bzw. die letzte richtige Zeile gelesen hat, gibt feof noch **nicht** true zurück, sondern erst nach dem nächsten Lesezugriff

Parsen von Optionen 1/3

UNI FREIBURG

- Oft will man in der Kommandozeile Optionen übergeben, zum Beispiel

  ```
  ./InputOutputMain --shift=3 --line-numbers MyFile.txt
  ```

  – Optionen sind auch erstmal ganz normale Argumente

  ```
  argv[0] :  ./InputOutputMain
  argv[1] :  --shift=3
  argv[2]:   --line-numbers
  argv[3] :  MyFile.txt
  ```

  – Da man das sehr oft hat, gibt es für das Parsing eine C-Bibliothek getopt, es reicht dafür am Anfang der Datei

  ```
  #include <getopt.h>
  ```

  – Siehe Codebeispiel gleich und man 3 getopt

Parsen von Optionen 2/3

UNI FREIBURG

- Kurze und lange Optionen
  - Ursprünglich waren Optionen einzelne Buchstaben und man schrieb nur ein Minus davor, und bei Optionen mit Argument kein Gleichheitszeichen

    ./InputOutputMain -s 3 -l MyFile.txt
  - Lange Optionen sind besser lesbar, aber dauern auch länger zu tippen
  - Deswegen ist es üblich, beides zu unterstützen bzw. zumindestens für die häufig benutzten Optionen einen Abkürzungsbuchstaben zu haben
  - Das geht mit **getopt_long**, siehe Codebeispiel gleich
  - Mit getopt_long werden die Argumente von links nach rechts, eins nach dem anderen abgearbeitet

Parsen von Optionen 3/3

UNI FREIBURG

- Zwei wichtige globale Variablen aus getopt.h
  - **optind** ist der Index von dem Argument, das getopt_long als nächstes bearbeit
    - optind ist mit dem Wert 1 initialisiert, aber Achtung, beim Testen hat man die getopt_long Schleife vielleicht mehrmals hintereinander, deswegen vorher immer optind = 1 setzen
  - **optarg** ist das Argument der zuletzt verarbeiteten Option, sofern sie ein Argument hatte (sonst NULL)
    - optarg ist vom Typ char*
  - Achtung: getopt_long ordnet die Argumente in argv um: nach dem Aufruf stehen die Argumente, die keine Optionen waren, am Ende
    - und optind ist der Index von dem ersten davon

Type casting, hier: const_cast

UNI FREIBURG

- Implizite Typkonvertierung
  - Manche Typkonvertierungen finden automatisch statt

```
bool flag = true;
int x = flag;  // Will get the value 1 (true = 1, false = 0).
```

- Explizite Typkonvertierung
  - Manchmal muss man einen Typ in einen anderen, eigentlich imkompatiblen, konvertieren

```
const char* fileName = "file.txt";
char** argv = new char*[2];
argv[1] = fileName;  // This will not compile.
argv[1] = const_cast<char*>(fileName);  // This will.
```

  - Es gibt auch noch static_cast, dynamic_cast und reinterpret_cast … siehe spätere Vorlesung (Vererbung)

Testen von Ein- und Ausgabe 1/3

UNI FREIBURG

- Testen einer Methode mit Eingabedatei
  - Den Test am Anfang eine Testdatei schreiben lassen
  - Und die dann am Ende wieder löschen
  - Siehe Codebeispiel
- Testen einer Methode mit Bildschirmausgabe
  - Option einführen, dass man auch in Datei schreiben kann
  - Einfach, weil die Bildschirmausgabe ja auch nur eine spezielle Datei ist ... nämlich stdout vom Type FILE*
  - Dann einfach Inhalt dieser Datei einlesen und schauen ob er korrekt ist
  - Testen von Benutzereingabe entsprechend

Testen von Ein- und Ausgabe 2/3

UNI FREIBURG

- Testen ob ein Programm "abbricht" wenn es soll
  - Zum Beispiel mit exit(1)
  - Das macht man mit ASSERT_DEATH

    ASSERT_DEATH(io.parseArguments(0, NULL), "Usage:.*");
  - Das erste Argument ist der Befehl der zu einem Programmabruch führen soll
  - Das zweite Argument ist ein regulärer Ausdruck der zu der Fehlermeldung passen muss, die dann kommt
  - **Wichtig dabei:** Diese Fehlermeldung muss nach stderr geschrieben werden, zum Beispiel so

    fprintf(stderr, "Usage: ./InputOutputMain ...\n");

Testen von Ein- und Ausgabe 3/3

UNI FREIBURG

- Besonderheiten bei der Benutzung
  - Die interne Realisierung von ASSERT_DEATH ist recht kompliziert, es wird ein sogenannter fork verwendet
    - Prozess wird in zwei Prozesse aufgespalten
  - Das gibt potenziell Probleme, wenn das Programm **threads** verwendet, also Teile hat, die unabhängig voneinander nebeneinander her laufen
  - Unsere Programme benutzen keine threads, und das müssen wir gtest mitteilen, sonst gibt es eine Warnung

    ::testing::FLAGS_gtest_death_test_style = "threadsafe";

    (das ist die Mitteilung, nicht die Warnung)

Literatur / Links

UNI FREIBURG

- fopen, fclose, feof, fgets, fprintf, fread, frwrite, ...
  - http://linuxmanpages.com/
  - Oder einfach man 3 fopen etc. in ein Terminal eingeben
- Parsen von Optionen
  - http://linuxmanpages.com/man3/getopt.3.php
  - Oder einfach man 3 getopt
- ASSERT_DEATH
  - http://code.google.com/p/googletest/wiki/GoogleTestAdvancedGuide#Death_Tests
- const_cast
  - http://www.cplusplus.com/doc/tutorial/typecasting