Programmieren in C++

SS 2012

Vorlesung 12, Mittwoch 24. Juli 2012

(Evaluation, Profiling, Arbeit am Lehrstuhl)

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 11. Übungsblatt (Projekt .h Dateien)
  - Dies ist die **letzte** Vorlesung ... Auswertung Evaluationsbögen
- Inhaltlich
  - Noch ein paar Infos zum Projekt
  - Und dann noch ein kleiner Exkurs zum Thema:
    - Wie bekomme ich meinen **Code schneller**?
    - Optimierungsflags, Profiling, Effizienz der STL
- Arbeit am Lehrstuhl
  - Arbeitsweise, aktuelle Projekte

Erfahrungen mit dem Ü11 (Projekt .h)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 24. Juli 13:45
  - Erstmal Schreiben der .h Dateien fanden die meisten gut
  - Aber auch nicht leicht vorherzusehen was gebraucht wird
  - Einige haben auch schon Teile implementiert
  - Diverse Fragen zum Projekt ... dazu gleich ein paar Folien
  - Punkte für Evaluationsbogen auf ÜB verteilen? ... nächste Folie
  - Wann volle Punktzahl im Projekt? ... optionaler Teil nicht nötig
  - Snake: wer verliert bei Berührung? ... der mit dem **Kopf** anstößt!
  - Wie wissen wir, ob Eval-Bogen wirklich hochgeladen? ... gar nicht
  - Vorschlag: besonders gute Übungsblätter auszeichnen

Punkte für den Evaluationsbogen

UNI FREIBURG

- Leicht abgeänderte Bonusregel
  - Manche haben in **allen** Übungsblätter fast alle Punkte
  - Denen bringt es dann nur ein paar wenige Punkte, für das schlechteste Übungsblatt 20 Punkte zu bekommen
  - Mehrere Anfragen dazu, deshalb folgende Alternativregel:

    +10 Punkte für das Projekt
  - Wir wenden **automatisch** die für Sie günstigere Regel an
  - Insbesondere: wenn jemand im schlechtesten Übungsblatt > 10 Punkte hat, sind die +10 für das Projekt günstiger
  - Sowieso ist dieser Kurs für (fast) alle eine **Studienleistung**
    - das heißt die Note zählt **nicht** zur Bachelor-Endnote
    - sie taucht aber in der Leistungsübersicht auf

Auswertung Evaluation 1/9

UNI FREIBURG

- Erstmal etwas Statistik ...
  - Zusammensetzung des Publikums nach **Studienfach**
    - 73 x BSc Informatik (fast alle im 2. Semester)
    - 14 x ESE
    - 1 x MST, 1 x Mathe, 1 x Bioinf, 1 x Lehramt, 1 x NF
  - Rücklauf (Stand: 24. Juli 13:40 Uhr)
    - Es sind noch 106 TeilnehmerInnen aktiv dabei
    - Es wurden 95 Evaluationsbogen abgegeben, also ≈ 90%
    - Lehrpreisvorschläge: 50, das sind über 50% ... **DANKE!**
  - Alle Details der Auswertung auf dem **Wiki**
    - Im Folgenden eine Zusammenfassung

Auswertung Evaluation 2/9

UNI FREIBURG

- Stil der Vorlesung
  - Atmosphäre: locker, angenehm, entspannt, ... (viele)
  - Dozentin sehr engagiert und motiviert (viele)
  - Dozentin erklärt sehr gut + unterhaltsam (viele)
  - Beste Vorlesung (einige)
  - Live Programmieren sehr lehrreich (viele)
    - Hin- und Herwechseln zwischen Folien, Editor und Konsolenausgabe am Anfang aber verwirrend (einige)
  - Statistik zu "Das inhaltliche Niveau der Veranstaltung ist"

    53 x hoch, 34 x angemessen, 1 x zu niedrig, 1 x zu hoch
  - Speziell am Anfang Tempo zu hoch (einige)
  - Seltener sagen "das ist ganz einfach" → "lose-lose" Situation

Auswertung Evaluation 3/9

UNI FREIBURG

- Inhalt der Vorlesung
  - Statistik zu "Ich habe in der Veranstaltung viel gelernt"

    53 x stimme voll zu, 27 x stimme zu, 8 x teils teils
  - Sehr praxisnah und lehrreich (viele)
  - "Low-level" Herangehensweise ist gut (einige)
  - Auch "Drumherum" war interessant / lehrreich (einige)
  - Themen für das Projekt interessant (einige)
  - Konsolen-Malen mochten einige irgendwann nicht mehr

Auswertung Evaluation 4/9

UNI FREIBURG

- Übungsblätter
  - Aufgaben interessant und lehrreich (viele)
    - Übungsblatt 10 (Vererbung) war das schlechteste
  - Aufgaben passen sehr gut zur Vorlesung (einige)
  - Schwierigkeit der Blätter schwankte stark (einige)
  - Statistik zu "Arbeitsaufwand für VL + Übung"

    33 x 5-8 Std, 39 x 9-12 Std, 14 x über 13 Std, 3 x 1-4 Std
  - Aufwand zu viel für 4 ECTS Punkte (einige)
  - Lag aber weniger an der Schwierigkeit der Aufgaben, sondern an dem "Drumherum", insbesondere Tests
  - Ab und zu musste man lästige Sachen selber rausfinden

    "Folien enthielten nur Bruchteile von dem was nötig war"

Auswertung Evaluation 5/9

UNI FREIBURG

- Materialien / Online-Angebot
  - Vorlesungsaufzeichnungen super + nützlich (sehr viele)
  - Statistik zu "Wie wurde die Vorlesung konsumiert"

    32 x Anwesenheit, 15 x Aufzeichnung, 38 x teils teils / beides
  - Forum super, Antworten kompetent und schnell (sehr viele)
  - Abgabesystem (Daphne, SVN, Jenkins) super (einige)
  - Statistik zu "Die Lehrmaterialien sind hilfreich"

    65 x stimme voll zu, 19 x stimme zu, 4 x teils teils

Auswertung Evaluation 6/9

UNI FREIBURG

- TutorInnen
  - Kommentare hilfreich (viele)
    - gerne noch mehr / weiterführend (einige)
  - Statistik zu "TutorIn kann gut erklären"

    21 x stimme voll zu, 32 x stimme zu, 17 x teils teils
  - Persönliches Treffen für einige hilfreich
  - "Tutorat per SVN und Forum wirkt etwas unpersönlich"
  - Persönlichen Kontakt zu Tutor früher herstellen
  - Diesmal keine Beschwerden wegen später Korrektur

Auswertung Evaluation 7/9

UNI FREIBURG

- Diverses
  - Nicht so viel Zeit mit Feedback verbringen (einige)
  - Manchmal die Zeit überzogen (einige)

    "Die Zeiteinteilung in der VL schien nicht vorhanden zu sein - sprich: es war teilweise extrem chaotisch!" (einer)
  - Nicht gefallen: (einer)

    "Die launische Dozentin, Literaturnähe hat gefehlt."

    "Lehrpreis: nein, um Himmels Willen, dann bekommt die ja schon wieder nen Preis"
  - Checkstyle zu streng mit "Line ends in whitespace" (einige)

    "Leerzeichen am Ende der Zeile sieht man nicht / hört man nicht"
  - Pause war gut / notwendig / schlecht / zu kurz (einige)
  - Zusätzliche Themen: Threads, GUIs, multidim. Arrays, ...

Auswertung Evaluation 8/9

UNI FREIBURG

- Verbesserungen gegenüber letztem Jahr
  - Übungsaufgaben weniger Arbeit + weniger unit tests
  - Weniger Vorgaben, letztes Jahr oft "einfach Abschreiben"
  - Vorlesungsaufzeichnungen noch besser geschnitten
  - Dozentin hat mehr und besser geschlafen
  - Zeitmanagement und Struktur weiter verbessert
  - Notenspiegel + Treffen mit Tutor im Vorfeld geklärt
  - Die Grundinfos dieses Mal vollständig(er) auf den Folien
  - Inhalt, Struktur, Reihenfolge weiter verbessert
  - Aufgabenstellung präziser (Musterlösungen vorher gemacht)
  - Beim Live-Programmieren nicht zuviel Code

Auswertung Evaluation 9/9

UNI FREIBURG

- Geplante Verbesserungen für nächstes Jahr
  - Vorkurs besser auf die Vorlesung abstimmen
  - Zeitmanagement weiter verbessern
  - Schauen, dass Feedback am Anfang nicht zu lange
  - Verhältnis Lerneffekt / Aufwand weiter optimieren
  - Übermäßig aufwändige Übungsblätter vermeiden
    - Unit tests nur an Stellen, wo essentiell und lehrreich
  - Übergang Folien / Code-Schreiben weiter optimieren
  - Früher persönlichen Kontakt zum Tutor anregen
  - Wieder etwas zu assert(...) sagen
  - Definitiv weniger "das ist ganz einfach" sagen

Hinweise zum Projekt 1/2

UNI FREIBURG

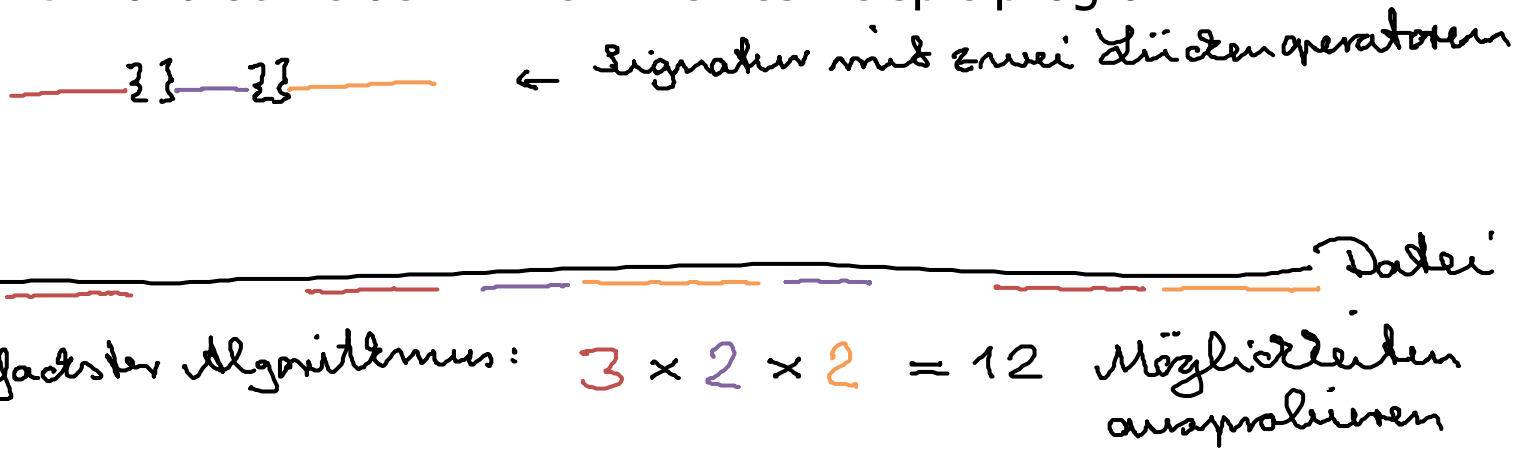
- Snake
  - Tastenbelegung via Kommandozeilenargument, z.B.

    ... --keys=65,66,67,68 [links, rechts, unten, oben]
  - Undefined reference ... bei Verwendung von ncurses
    - beim Linken -lncurses dazu schreiben
  - Bei allen weiteren Fragen:
    - Post an das **Forum** (wird auch andere interessieren)
    - Oder Mail an den Tutor (aber besser Forum)
    - Auf Fragen in Ihren letzten erfahrungen.txt sollten Sie eine Antwort von Ihrem/r Tutor/in bekommen

Hinweise zum Projekt 2/2

UNI FREIBURG

- Virenscanner Algorithmus
  - Für die Lückenoperatoren, muss man
    - die Signatur in "Stücke" zerlegen
    - alle Vorkommen dieser Stücke finden
    - alle Kombinationen ausprobieren (ob die Lücken passen)
  - Problem: wie probiert man alle Kombinationen aus?
    - Dazu schreiben wir ein kleines Beispielprogramm



Performance / Profiling

UNI FREIBURG

- Fragen dabei
  - Läuft mein Programm so schnell wie es könnte?
  - Und wenn nein, wie kann ich es schneller machen?
- Es gibt im Wesentlich drei Potenziale
  - **Algorithmische Verbesserung**

    Zum Beispiel von dem Algorithmus zur Realisierung der Lückenoperatoren auf der Folie vorher
  - **Algorithm Engineering**

    Wissen darüber, welche Konstrukte im Code aus welchen Gründen wie lange dauern
  - **Compileroptimierung**

    Wissen darüber, wofür der Compiler unter welchen Umständen welchen Maschinencode erzeugt

Beispielprogramm

UNI FREIBURG

- Schreiben eines großen Arrays
  - Ein einfaches aber in vielerlei Hinsicht typisches Beispiel
    - Große Datenmengen
    - Verarbeitung in einer Schleife
    - Jede Iteration macht etwas relativ Einfaches
- Effekte die im Folgenden eine Rolle spielen
  - Kosten für Speicherallokation
  - Kosten für Funktionsaufrufe
  - Compileroptimierung ...
    - für möglichst einfachen Maschinencode
    - für Maschinencode, der Besonderheiten der CPU nutzt

Kosten für Speicherallokation

UNI FREIBURG

- Speicherallokation hat Kosten
  - ... und zwar fixe Kosten pro Allokation + Kosten linear in der Größe des allozierten Speichersegmentes
  - Deswegen sollte man vermeiden
    - unnötig viele kleine Speicherallokationen
    - unnötige große Speicherallokationen
  - In unserem Beispiel können wir den Speicher für das Ausgabefeld **vorher** allozieren, weil wir vorher schon wissen wie groß das Feld wird
    - spart Reallokationen (Umkopieren)
    - spart Overhead bei der Allokation (siehe oben)

Kosten für Funktionsaufrufe 1/2

UNI FREIBURG

- Funktionsaufrufe haben Kosten
  - Im Maschinencode muss der lokale Kontext und die aktuelle Adresse im Programmcode gesichert werden
  - Dann müssen die Argumente kopiert werden
  - Dann muss zu der Adresse der Funktion gesprungen werden
  - Am Ende der Funktion muss wieder zurückgesprungen und der lokale Kontext wieder hergestellt werden
  - Bei Funktionen, in denen verhältnismäßig viel passiert sind diese Kosten vernachlässigbar
  - Aber bei Funktionen, die nur sehr einfache Dinge tun, können diese Kosten größer sein als die des eigentlichen Codes der Funktion

Kosten für Funktionsaufrufe 2/2

UNI FREIBURG

- Vermeiden von Funktionsaufrufen
  - Wenn man eine Funktion in der .h Datei implementiert und sie nicht zu groß ist, wird der Compiler den Funktionsaufruf einsparen
  - ... indem er einfach den Code der aufgerufenen Funktion an die Stelle des Aufrufes setzt
  - Insbesondere passiert das bei den ganzen "kleinen" Funktionen aus der STL wie std::vector::size, std::vector::push_back, usw.
  - Allerdings nur mit der Compileroption -O, siehe gleich

Profiling

UNI FREIBURG

- Wo verbraucht mein Programm wieviel Zeit?
  - Das erfährt man mit einem sogenannten **profiler**
  - Zum Beispiel gprof, das ist der GNU Profiler
  - Einfach das Programm mit der Option –pg übersetzen

    CXX = g++ -pg ...
  - Dann das Programm normal bis zum Ende laufen lassen, das erzeugt eine Datei gmon.out
  - Die erhält Informationen darüber wie oft sich das Programm in welchem Teil des Codes aufgehalten hat
  - Man schaut sich aber nicht die rohe Datei gmon.out an, sondern ruft gprof mit dem Namen der ausf. Datei auf

    gprof ./ArrayFillMain

Maschinencode 1/2

UNI FREIBURG

- Den erzeugten Maschinencode
  - ... kan man sich mit g++ einfach wie folgt anschauen

    g++ -S ArrayFillMain.cpp
  - Das erzeugt dann eine Datei

    ArrayFillMain.s
  - Das ist quasi der Maschinencode der .o Datei in menschenlesbarer Form, in sog. **Assembler**sprache
  - Die (triviale) Übersetzung in Maschinencode geht dann mit

    g++ -c ArrayFillMain.s

Maschinencode 2/2

UNI FREIBURG

- Optimierter Code
  - Ohne Optimierung erzeugt g++ Maschinencode der **eins zu eins** dem C/C++ Code entspricht **...** siehe Codebeispiel
  - Mit Optimierung wird versucht, Code zu erzeugen, der schneller läuft, dabei gibt es verschiedene Stufen

    g++ -O1 ...

    g++ -O2 ...

    g++ -O3 ...
  - Details zu was da genau optimiert wird: siehe Referenzen
  - Als Faustregel gilt: je höher die Optimierungsstufe, desto **größer** der Code und desto **schneller** läuft er
  - Die Option -O1 bringt aber schon das meiste

Aktuelle Forschung

UNI FREIBURG

- Forschung an unserem Lehrstuhl
  - Wir machen Algorithmen und Datenstrukturen
    - 1/3 Theorie (neue Algorithmen, Laufzeitanalyse, etc.)
    - 1/3 Algorithm Engineering (gute Implementierungen)
    - 1/3 Software Engineering (gute Software)
  - Aktuelle Projekte
    - Routenplanung, insbesondere die auf Google Maps
    - Suchmaschinen, insbesondere CompleteSearch & Broccoli
  - Aktuelle Arbeiten dazu
    - http://ad.informatik.uni-freiburg.de/papers

Literatur / Links

UNI FREIBURG

- gprof
  - http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html
- g++ optimization levels
  - http://www.network-theory.co.uk/docs/gccintro/gccintro_49.html
  - http://linuxmanpages.com/man1/g++.1.php
  - oder einfach man g++
- Loop unrolling
  - http://en.wikipedia.org/wiki/Loop_unwinding