Objektorientierte Programmierung

(Winter 2010/2011)

Prof. Dr. Stefan Brass
Institut für Informatik

Übungsleiterin: Dipl.-Inform. Annett Thüring

Tutorium: Dr. Henning Thielemann

Inhalt

Lernziele

- Programmieren können

  Selbständige Erstellung eigener Programme für gegebene Aufgaben.
- Rechner, Betriebssystem und Editor/Entwicklungsumgebung ausreichend bedienen können.
- Gegebene Programme verstehen können

  Z.B. Klausurfrage: Was gibt dieses Programm aus?
- Die syntaktische Korrektheit gegebener Programme beurteilen können (→ Syntax-Formalismen).
- Grundkonzepte von Programmiersprachen kennen, sich leicht in neue Sprachen einarbeiten können.

Motivation (1)

Warum sollten Sie programmieren lernen?

- grundlegende handwerkliche Fähigkeit, im Berufsleben eines Informatikers notwendig.

  Selbst wenn Sie nicht an der Entwicklung wirklich großer Programme mitwirken, ist Programmierung z.B. auch nötig, um Inhalte von Datenbanken schön aufbereitet ins Netz zu stellen.
- komplexere Software-Werkzeuge bieten für Nicht-Standard-Aufgaben meist Möglichkeiten, die ähnlich zu einer Programmiersprache sind.

  Z.B. ist die Datenbanksprache SQL zwar keine allgemeine Programmiersprache, aber hat doch vieles gemeinsam mit Programmiersprachen. Programmierkenntnisse erleichtern das Erlernen von SQL.

Motivation (2)

Warum sollten Sie programmieren lernen? (Forts.)

- Vieles in der Informatik kann man besser verstehen, wenn man es im Prinzip auch selbst programmieren könnte.
- Es erleichtert Verhandlungen mit Programmierern, wenn man auch selber programmieren kann.
- Meine Kollegen und ich sind der Meinung, dass niemand ein Diplom/Bachelor-Abschluß in Informatik, Bioinformatik, oder Wirtschaftsinformatik bekommen sollte, der nicht programmieren kann.

Motivation (3)

Warum macht Programmieren Spaß?

- Man kann mit relativ wenig Aufwand komplexe Maschinen und virtuelle Welten (Spiele) konstruieren.

  Oder reale Maschinen / elektronische Schaltungen (Microcontroller).
- Man kann sich das Leben im Umgang mit dem Computer etwas erleichtern.

  Stupide manuelle Tätigkeiten durch kurzes Programm automatisieren.
- Den Computer beherrschen statt umgekehrt.

  Es ist nett, Computer-Experte zu sein. Dazu gehören Programmierkenntnisse, aber auch Hardware- und Betriebssystem-Kenntnisse.
- Konkrete Anwendung von Mathematik.

Motivation (4)

Warum ist Programmieren eine Herausforderung?

- Computer befolgen genau die gegebenen Regeln (das Programm), aber können kein bißchen selbst denken (nichts versteht sich von selbst).
- Man muß bei der Programmierung also an alles denken: Alle möglichen Fälle, auch Ausnahmen.

  Ein Programm ist wie ein mathematischer Beweis, vielleicht sogar hinsichtlich der Präzision noch anspruchsvoller. Man kann kein Glied in der Kette auslassen, es gibt nichts, "was sich von selbst versteht".
- Hier muß man Perfektionist sein.
- Man muß sich in die Maschine hineindenken.

Motivation (5)

Ich will programmieren lernen. Was brauche ich?

"Zunächst einmal brauchen Sie wie beim Erlernen einer Fremdsprache sehr viel Eifer — aber zusätzlich noch Freude an kreativer Arbeit und ein ausgeprägtes Abstraktionsvermögen. Letzteres bedeutet, Probleme soweit aufdröseln zu können, dass man die daraus entstehenden Teilprobleme direkt als Programm formulieren kann. Der kreative Anspruch ergibt sich daraus, dass es häufig mehrere Wege zur Lösung gibt, etwa besonders elegante oder besonders kurze — oder vermeidbar umständliche." [Oliver Lau: FAQ Programmieren lernen. c't 2008/23.]

Zur Programmiersprache (1)

- In dieser Vorlesung wird Programmierung anhand der Sprache "C++" gelehrt.

  Gelegentlich auch Vergleich mit anderen Sprachen (z.B. Pascal, Java).

- C++ ist:
  - ◇ in der Praxis verbreitet, seit Jahren bewährt.

    Insbesondere auch häufig in Stellenanzeigen verlangt.

  - ◇ gut mit Software-Werkzeugen unterstützt.

    Es gibt z.B. eine ganze Reihe von Compilern (Übersetzer von C++ in die Maschinensprache der CPU im Rechner) unterschiedlicher Hersteller, und auch einen verbreiteten und guten Open Source Compiler (g++). Es gibt einen internationalen Standard, der nicht an eine spezielle Firma gebunden ist.

Zur Programmiersprache (2)

- C++ ist (Forts.):
  - ◇ recht hardware-nah, effizient implementierbar.

    Ich meine, dass zum Verständnis einer Programmiersprache auch eine Vorstellung davon gehört, wie die Programme auf realer Hardware ausgeführt werden. Hardwarenahe Sprachen sind auch besonders geeignet für Anwendungen, bei denen es sehr auf die Leistung ankommt. Hardwarenähe erleichtert auch den späteren Einstieg in die Embedded-Programmierung (Waschmaschinen etc.).

  - ◇ eine umfangreiche Multiparadigma-Sprache.

    Eine Programmierparadigma (Sammlung von Konzepten zur Programmierung) wäre die objektorientierte Programmierung. Die vielfältigen Möglichkeiten von C++ erlauben, dass viele Konzepte hier besprochen werden können, was den späteren Umstieg auf andere Sprachen erleichtert. Man kann allerdings schon in einer C++-Teilmenge gut programmieren.

Zur Programmiersprache (3)

Auf die genaue Sprache kommt es nicht an:

- Wenn man Programmiersprachen mit Fremdsprachen vergleicht, sind Wortschatz und Syntax um ein Vielfaches einfacher.

  Der wesentliche Aufwand beim "Programmieren lernen" sind nicht so sehr die spezifischen Konstrukte einer Programmiersprache, als zu lernen, sich in Computer hineinzudenken, und einen Schatz von Mustern für bestimmte Aufgaben im Kopf zu haben, die man nach Bedarf aktivieren und anpassen kann. Diese Muster sind aber in weiten Grenzen unabhängig von der Programmiersprache, außer bei ganz anderen Programmierparadigmen, wie etwa logischer Programmierung.

- Sie werden noch viele andere Programmiersprachen lernen müssen.

Zur Programmiersprache (4)

- Wenn Sie C++ beherrschen, können Sie z.B. Java (den Kern, die Sprache selbst) in einem Tag lernen.

  Die beiden Sprachen sind recht ähnlich, es reicht also, sich auf die Unterschiede zu konzentrieren. Zum großen Teil sind das Dinge, die es in C++ gibt, und die in Java irgendwie automatisch funktionieren. Insofern wird der Umstieg von C++ nach Java einfacher sein als in der umgekehrten Richtung. Natürlich wird es mehr als einen Tag Übung benötigen, um Java wirklich flüssig zu schreiben.

- Was dann noch hinzu kommt, sind die Bibliotheken (vordefinierte Klassen und Funktionen).

  Das kann man aber nach Bedarf lernen. In dieser Vorlesung ist ohnehin wenig Zeit für Bibliotheksfunktionen in C++, dafür wäre also relativ wenig Zeit "verschwendet", wenn Sie eigentlich Java lernen wollen.

Inhalt

Ansprechpartner (1)

Dozent: Prof. Dr. Stefan Brass

- Email: `brass@informatik.uni-halle.de`

  Betreff-Zeile sollte mit `[oop10]` beginnen, möglichst aussagefähig.
- Büro: Von-Seckendorff-Platz 1, Raum 313.
- Telefon: 0345/55-24740.
- Sprechstunde: Dienstags, 12^{15}–13^{15}.
- Frühere Unis: Braunschweig, Dortmund, Hannover, Hildesheim, Pittsburgh, Gießen, Clausthal.
- Spezialgebiet: Datenbanken, Wissensrepräsentation.

  Oracle8 Certified DBA. IBM Certified Advanced DBA (DB2 8.1).

Ansprechpartner (2)

Übungsleiterin: Dipl.-Inform. Annett Thüring

- Büro: Von-Seckendorff-Platz 1, Raum 319.
- Sprechstunde: Mittwochs, 14^{00}–15^{00}.
- Telefon: 0345/55-24739.
- Email: `thuering@informatik.uni-halle.de`

Tutorium: Dr. Henning Thielemann

- Büro: Von-Seckendorff-Platz 1, Raum 314.
- Telefon: 0345/55-24773.
- Email: `oop10@henning-thielemann.de`

Ansprechpartner (3)

Sekretärin: Ramona Vahrenhold

- Büro: Von-Seckendorff-Platz 1, Raum 324.
- Telefon: 0345/55-24750, Fax: 0345/55-27333.
- Email: `vahrenhold@informatik.uni-halle.de`
- Das Sekretariat ist dienstags nicht besetzt.

Rechnerbetriebsgruppe:

- Poolaufsicht (studentische Hilfskräfte): Raum 333.
- Daniel Trull, Lutz Ohme: Raum 320.

Webseiten

- `http://www.informatik.uni-halle.de/~brass/oop10/`
  - ◇ Folien.

    Die Folien werden jeweils vor der Vorlesung ins Web gestellt.
  - ◇ Alte Klausuren und Programmiertests.
  - ◇ Nützliche Links.
- `http://studip.uni-halle.de`

  `.../details.php?sem_id=cb795ac2352572b8876b1b70a97b5dc2`
  - ◇ Anmeldung zu Vorlesung und Übungsgruppen.
  - ◇ Link zu Übungsplattform (Hausaufgaben).
  - ◇ Forum (z.B. auch Fragen zu Hausaufgaben).

Zeit und Ort (1)

Vorlesung (2 SWS):

- Dienstags, 10^{15}–12^{00} (15 min Pause), Raum 3.28.

Übung am Rechner (2 SWS):

- Sieben Übungsgruppen, je max. 25 Teilnehmer:

Nr	Tag	Zeit	Beginn
1 (3.34) / 4 (3.32)	Montag	12^{15}–13^{45}	4.10.
2 (3.34) / 5 (3.32)	Montag	14^{15}–15^{45}	4.10.
3 (3.34) / 6 (3.32)	Montag	16^{15}–17^{45}	4.10.
7 (3.34)	Dienstag	14^{15}–15^{45}	5.10.

Falls erste Übung (Linux-Einführung) verpasst: Mittwoch, 16-18, 3.32 oder erstes Tutorium am Dienstag, 16^{00}-17^{30}, 3.32.

Zeit und Ort (2)

Anmeldung zu Übungsgruppen:

- Wir garantieren, dass Sie einen Platz in einer der Übungsgruppen bekommen.

  Wenn Sie zur Teilnahme an dieser Vorlesung verpflichtet sind. Notfalls werden zusätzliche Gruppen angeboten (zu anderen Zeiten).
- Wir können leider nicht garantieren, dass Sie einen Platz in einer bestimmten Gruppe bekommen.
- Sie müssen sich im Web über StudIP für eine Gruppe eintragen: [http://studip.uni-halle.de/].

  Direkte Links: [http://www.informatik.uni-halle.de/~brass/oop10/].

Zeit und Ort (3)

Anmeldung zu Übungsgruppen, Forts.:

- Für die Anmeldung zu einer Übungsgruppe über StudIP brauchen Sie einen Benutzernamen und ein Passwort/Kennwort, das Ihnen mit der Immatrikulationsbescheinigung zugegangen sein müßte.

  Der Benutzername identifiziert Sie eindeutig im System (Vor- und Nachname sind nicht immer eindeutig). Das Passwort sollten nur Sie wissen. Es dient als Ausweis, dass Sie auch wirklich Sie sind.
- Bitte tragen Sie sich bei StudIP nicht nur für die Übungsgruppe Ihrer Wahl, sondern auch für die Vorlesung selbst ein.

Übersicht

Um dieses Modul erfolgreich abzuschließen, müssen Sie

- ☐ sich bei StudIP für eine Übungsgruppe anmelden,
- ☐ sich zum Modul anmelden (Löwenportal),
- ☐ Hausaufgaben bearbeiten und abgeben,
- ☐ in den Übungen ausreichend aktiv mitarbeiten,
- ☐ den praktischen Programmiertest bestehen,
- ☐ sich zur Prüfung anmelden (Löwenportal),
- ☐ die Prüfung (Klausur) bestehen.

Modulanmeldung

- Für fast alle Studiengänge ist die Modulanmeldung über das Löwenportal Pflicht.

  [http://www.verwaltung.uni-halle.de/SBStud/]
  Anleitung im grünen Kasten rechts unter Weitere Einstellungen/Hilfe.
  Falls über Löwenportal nicht möglich, dann im Prüfungsamt.

- Die Modulanmeldung ist nur bis zum 31.10.2010 möglich.
- Die Modulanmeldung ist zwingende Voraussetzung für die spätere Anmeldung zur Prüfung.

  Ein Zweck der Modulanmeldung ist die Prüfung von Voraussetzungen, dieses Modul hat aber keine Voraussetzungen.

Inhalt

Tutorium (1)

- Diese Vorlesung wird von Studierenden mit sehr unterschiedlichen Vorkenntnissen besucht.
  - ◇ Einige können schon programmieren, vielleicht sogar in C++.

    Bedenken Sie aber bitte, dass diese Vorlesung auch Konzepte vermitteln soll, die in einer rein praktischen Ausbildung fehlen.

  - ◇ Für andere Teilnehmer ist die Programmierung ein völliges Neuland.
- Mein Anspruch ist, dass auch die zweite Gruppe der Vorlesung folgen kann: Ich möchte den Stoff "von Grund auf" logisch und vollständig präsentieren.

Tutorium (2)

- Stellen Sie Fragen, wenn Sie etwas nicht verstehen, gerne auch während der Vorlesung.

  Wer schon so lange programmiert wie ich, übersieht manchmal, dass bestimmte Dinge nicht selbstverständlich sind. Ich bin dankbar für Hinweise, die es mir erlauben, mein Skript (Foliensatz) noch zu verbessern. Andere Studierende werden Ihnen auch dankbar sein.

- Die Erfahrung zeigt, dass
  - ◇ manche Programmier-Neulinge sich gut in Maschinen und Formalismen hineindenken können und die Programmierung ohne Probleme lernen,
  - ◇ andere aber sich schwerer tun.

Tutorium (3)

- Für diejenigen Studierenden, die mit dieser Vorlesung Probleme haben, gibt es ein Tutorium:
  - ◇ Treffen: Dienstags, 16^{00}–17^{30}, Raum 3.32.
  - ◇ Beginn: 5.10., Anmeldung über StudIP.
  - ◇ Trainer: Dr. Henning Thielemann.
  - ◇ EMail: `oop10@henning-thielemann.de`
- Scheuen Sie sich nicht, von diesem Angebot Gebrauch zu machen.

  Offiziell fordert diese Vorlesung keine Vorkenntnisse, es ist also ok, wenn Sie keine haben. Es zeugt von Klugheit, Probleme rechtzeitig zu erkennen, und Hilfe von Fachleuten anzunehmen.

Tutorium (4)

- Im Tutorium gibt es nur ungefähr 25 Plätze.
- Die Erfahrung zeigt, dass nur ein kleiner Teil der Studierenden, die Probleme mit dieser Vorlesung haben, das Tutorium besucht.

  Daher werden die Plätze reichen. Die Erfahrung zeigt auch, dass regelmäßige Teilnehmer am Tutorium die Vorlesung am Ende bestehen.

- Das Tutorium ist nur offen für Studierende, die mit der Programmierung noch nicht zurecht kommen.

  Oft besuchen gerade besonders engagierte Studenten das Tutorium, die glauben, dort interessanten Zusatzstoff vermittelt zu bekommen. Dafür ist es nicht gedacht. Sie würden schwächeren Studierenden Betreuungskapazität wegnehmen und das Klima im Tutorium stören.

Tutorium (5)

- Es gibt außerdem ein Tutorium speziell für Frauen.

  Es wird gehalten von Sarah Scharfenberg (fortgeschrittene Studentin, außerdem Leiterin einer Übungsgruppe). Das Tutorium wird finanziert vom Büro der Gleichstellungsbeauftragen. Falls Sie Interesse an einer Teilnahme haben, wenden Sie sich bitte an die Leiterin dieses Tutoriums [sarah.scharfenberg@student.uni-halle.de].

  Selbstverständlich sind männliche und weibliche Studierende gleichermaßen willkommen im Tutorium von Dr. Thielemann.

- Falls Sie schon gut programmieren können, und bereit wären, gegen Erlass einiger Hausaufgaben einige Stunden in das "Coachen" von Mitstudierenden zu investieren, wenden Sie sich an Dr. Thielemann.

  Natürlich auch, wenn Sie umgekehrt Interesse an solcher Hilfe haben.

Inhalt

Hausaufgaben (1)

- Es gibt wöchentlich Hausaufgaben (zum größten Teil Programmieraufgaben).
- Die Aufgaben werden immer Dienstag abends ins StudIP gestellt.
- Sie sind dann bis zum Dienstag der nächsten Woche (24:00) über die Übungsplattform im StudIP abzugeben.

  Programme bitte als unter Linux mit g++ compilierbarer Quellcode (ASCII), nicht als Word oder PDF-Dateien. Für theoretische Aufgaben wird ASCII oder PDF akzeptiert.

Hausaufgaben (2)

- Die Aufgaben sind einzeln zu bearbeiten.

  Eigentlich wäre Gruppenarbeit gut, wenn alle Mitglieder schon mit einer eigenen Lösung zum Treffen kommen, und man dann die Ansätze vergleicht, und aus den verschiedenen Ideen gemeinsam eine "noch perfektere" Lösung konstruiert. Leider läuft es öfters so, dass einer alles macht, und die anderen ihre Namen mit draufschreiben. Das wäre gefährlich (einer besteht die Klausur, die anderen nicht). Deswegen können wir leider keine Gruppenarbeit erlauben.

- Zu ähnliche (abgeschriebene) Abgaben werden mit 0 Punkten für alle Beteiligten bewertet.

  In nicht ganz klaren Fällen werden Sie Ihre Abgabe in der Übung vor der ganzen Gruppe erklären müssen, oder zu einem mündlichen Test beim Professor bestellt. Es gibt Plagiats-Erkennungssoftware.

Hausaufgaben (3)

- Nur Hausaufgaben, die auf dem Aufgabenzettel mit Punkten gekennzeichnet sind, werden von den Tutoren (Studenten aus höherem Semester) korrigiert.

  Aus Kapazitätsgründen können wir leider nicht alle korrigieren.

- Falls Sie die Korrektur oder die Bepunktung nicht verstehen, fragen Sie bitte.

  Wenden Sie sich bitte zuerst an Ihren Tutor, und dann an die Übungsleiterin oder den Professor. Fehler kommen leider gelegentlich vor. Selbst wenn sich am Ende herausstellt, dass der Punktabzug berechtigt war, haben bei der Diskussion beide Seiten etwas gelernt. "Wer nicht fragt, bleibt dumm."

- Sie müssen mindestens 50% der Punkte erreichen.

Hausaufgaben (4)

- Die unbepunkteten Hausaufgaben müssen auch abgegeben werden.

  Wer diese Aufgaben nicht abgibt, verliert einen Punkt pro Aufgabe.

- Der Tutor wählt Übungsteilnehmer aus, die ihre Aufgabe in der Übung präsentieren.

  Wer seine Abgabe für unbepunktete Aufgaben nicht erklären kann: 10 Minuspunkte. Wer die bepunkteten Aufgaben nicht erklären kann, verliert alle Punkte der Aufgabe und bekommt eine "gelbe Karte".

- Man darf in der Übung bis zu drei Mal fehlen.

  Fehlen Sie mehr als drei Mal, können Sie 0 Punkte für Hausaufgaben bekommen, die Sie (wegen Abwesenheit) nicht erläutern konnten (Ermessen der Übungsleiterin).

Hausaufgaben (5)

- Übliche Regeln für guten Programmierstil müssen eingehalten werden:
  - ◇ Sinnvolle Formatierung.

    Zeilenumbrüche, Einrückungen ähnlich wie in der Vorlesung.
  - ◇ Ausreichend Kommentare.
  - ◇ Keine üblen Programmiertricks (z.B. kein `goto`).
- Kann der Tutor Ihr Programm nach ca. 5 Minuten noch nicht verstehen, bekommen Sie 0 Punkte.
- Wenn Ihre Abgabe nicht mit `g++` unter Linux compilierbar ist, bekommen Sie 0 Punkte.

Übung

- In der Übung werden die Hausaufgaben (bepunktet und unbepunktet) besprochen.

  Nutzen Sie die Gelegenheit, Alternativen zu Ihrer Lösung anzuschauen. Auch interessante Fehler werden besprochen. Bringen Sie interessante Aspekte Ihrer Lösung aktiv ein, nicht nur nach Aufforderung.

- Sie können Fragen in einer kleineren Gruppe stellen.

  Sie können Fragen selbstverständlich auch in der Vorlesung stellen.

- Möglichkeit zum Erfahrungsaustausch.
- Lösen weiterer Aufgaben (Präsenzaufgaben) ohne Bewertung, aber mit Hilfe/Feedback vom Tutor.
- Fragen zu den anstehenden Hausaufgaben.

Programmiertest (1)

- Die Modulvorleistung (Voraussetzung für die Zulassung zur Klausur) besteht aus:
  - ◇ Erreichen von 50% der Hausaufgabenpunkte,
  - ◇ Bestehen von 2 von 4 Programmiertests (praktische Prüfung am Rechner).

    Wer die Hausaufgaben selbst bearbeitet hat, und wenigstens hinterher Aufgaben von diesem Typ ohne fremde Hilfe lösen kann, sollte mit den Programmiertests keine Schwierigkeiten haben. Leider gab es in der Vergangenheit "schwarze Schafe", die sich bei den Hausaufgaben durchgemogelt haben. Wir wollen aber mit Erreichen der Modulvorleistung wenigstens ein Minimum an praktischen Programmierfähigkeiten garantieren können.

Programmiertest (2)

- Die Programmiertests findet an drei Übungsterminen statt, und einem Termin in den Semesterferien:
  - ◇ 08./09.11.2010 (Montags/Dienstags-Übung)
  - ◇ 13./14.12.2010
  - ◇ 24./25.01.2011
  - ◇ 07.02.2011 (Mo, Termin wird ggf. verschoben).
- Wenn alles gut verläuft, brauchen Sie zum dritten und vierten Termin gar nicht mehr zu erscheinen.

  Sie dürfen dann auch nicht teilnehmen, damit wir uns den restlichen Kandidaten mit mehr Aufmerksamkeit widmen können.

Programmiertest (3)

- Beim Programmiertest müssen Sie eine Funktion oder Klasse in einem gegebenen Rahmenprogramm schreiben.

  Das Rahmenprogramm enthält Test-Aufrufe Ihrer Funktionen.
- Sie müssen das Programm innerhalb von ca. 60 Minuten zum Laufen bekommen, so dass es die erwarteten Ausgaben macht.

  Natürlich muß Ihr Programm nicht nur für die Beispielaufrufe funktionieren, sondern insgesamt korrekt sein. Es gibt nur "bestanden" oder "nicht bestanden". Unfertige oder nicht richtig funktionierende Programme werden nicht akzeptiert. Bei verdeckten Fehlern, die von unseren Tests nicht erfasst werden, werden wir eher großzügig sein.

Programmiertest (4)

- Bestehen Sie die Modulvorleistung (Hausaufgaben und Programmiertest) nicht, so können Sie diese Vorlesung nur ein Jahr später erneut belegen.

  Es wird dann eine andere Programmiersprache (C#) unterrichtet. Bestimmte Module des 2. und 3. Semesters können Sie dann auch nicht belegen (weil sie OOP bzw. die Modulvorleistung als Voraussetzung haben). Bei besonderen Belastungen (längerer Krankheit etc.) wenden Sie sich an den Dozenten wegen möglicher Ersatzleistungen.
- Sollten Sie also den ersten Programmiertest nicht bestehen, werden Sie bitte aktiv.
- Trost: Wenn Sie nicht zur Klausur zugelassen werden, haben Sie keinen Prüfungsversuch verbraucht.

Prüfung (1)

- Die Prüfung erfolgt als Klausur.
- Termin: 22.03.2011, 10^{00}-12^{00}, Raum 5.09 u.a.

  Achten Sie aber zur Sicherheit auf weitere Ankündigungen.
- Man muß sich bis vier Wochen vorher zur Klausur anmelden.

  Voraussetzung: zum Modul angemeldet und Modulvorleistung erfüllt.
- Man kann sich bis drei Tage vorher ohne Angabe von Gründen wieder abmelden.

  Alle Angaben zur Prüfungsordnung sind ohne Gewähr (möglicherweise auch Studiengangs-abhängig). Informieren Sie sich bitte selbst beim Prüfungsamt oder dem Studienberater.

Prüfung (2)

- Bücher, eigene Notizen, Kopien sind erlaubt.

  Bücher sind nur nützlich, wenn man sie vorher gelesen hat. Die Zeit bei der Klausur ist begrenzt, man muß also wissen, wo man etwas suchen muß. Sich einen "Spickzettel" oder "Quick Reference" zu machen, kann eine gute Prüfungsvorbereitung sein.

- Rechner aller Art sind nicht erlaubt.

  Sie können also zu entwickelnde Programme nicht ausprobieren. Wenn man vorher nur mit langem Herumprobieren und viel Feedback vom Compiler ein funktionierendes Programm entwickeln konnte, ist man noch nicht reif für die Klausur. Üben Sie auch das Programmieren auf dem Papier und anschließendes Eingeben in den Rechner. Man denkt wahrscheinlich ohnehin besser mit etwas Abstand vom Rechner.

Prüfung (3)

- Die Note für das Modul basiert nur auf der Klausur.

  Der Dozent behält sich das Recht vor, in seltenen Ausnahmefällen Extrapunkte (bis max. 5% der Klausurpunkte) zu vergeben für das Finden von Fehlern im Skript, außergewöhnlich gute Hausaufgabenergebnisse verbunden mit entsprechend aktiver Übungsteilnahme, oder eventuell andere Aktivitäten, die allen Teilnehmern zugute kommen.

- Es wäre nicht klug, mit den Hausaufgaben vorzeitig aufzuhören (wichtige Prüfungsvorbereitung).
- Falls Sie 60% der Klausurpunkte erreicht haben, haben Sie garantiert bestanden.

  Ab 95% gibt es die bestmögliche Zensur (1.0, A). Die Grenzen werden möglicherweise noch nach unten verschoben.

Prüfung (4)

- Falls Sie an der Klausur teilnehmen und bestehen, gibt es keine Möglichkeit zur Zensurverbesserung.
- Falls Sie bei der Klausur durchfallen, gibt es noch eine Wiederholungsmöglichkeit.

  Wenn Sie nicht erscheinen, obwohl Sie angemeldet sind, und keine Krankschreibung vorlegen, sind Sie automatisch durchgefallen. Es ist auch möglich, sich erst zum 2. Termin anzumelden. Dann gibt es aber keine Wiederholungsmöglichkeit im Rahmen dieser Vorlesung mehr.

- Nachholtermin: voraussichtlich 19.07.2011, 10-12.

  Bitte informieren Sie sich über Ihre Prüfungsordnung, inwieweit eine extra Anmeldung nötig ist, oder Sie automatisch angemeldet sind, falls Sie die erste Klausur nicht bestanden haben.

Prüfung (5)

- Falls Sie die Nachholklausur auch nicht bestehen, müssen Sie das Modul ein Jahr später neu belegen.

  Der andere Professor und die andere Programmiersprache sind auch eine Chance für einen Neuanfang.

- Es gibt maximal drei Prüfungsversuche.

  Nach dem dritten nicht erfolgreichen Prüfungsversuch für diese Vorlesung ist für Informatiker und Wirtschaftsinformatiker das Studium beendet. Im Laufe des Studiums sind nur 7 bzw. 8 zweite Wiederholungsprüfungen erlaubt, eventuell nur mit Antrag. Endgültiges Durchfallen (nach drei Prüfungsversuchen) reduziert Ihre Optionen zum Studienfachwechsel deutlich (z.B. Wirtschaftsinformatik → Wirtschaftswissenschaften ist dann nicht mehr möglich). Lassen Sie sich vor dem letzten Prüfungsversuch unbedingt kompetent beraten.

Inhalt

Rechnernutzung, Software (1)

Rechnerpools:

- PC-Pool: Raum 332

  29 PCs (Athlon64 X2 4200+, 1 GB RAM, 19" TFT-Monitor).
  Windows XP Professional / Ubuntu Linux 10.04 64bit

- ThinClient-Pool: Raum 334

  Windows Server "odin" (4x Opteron 852, 16 GB, Windows 2003)
  Linux Server "anubis" (4x Opteron 852, 16 GB, Ubuntu 8.04 64bit)
  Unix Server "turing" (4x Sparc II 400 Mhz, 2 GB, Solaris 10 sparc)

- Server für Home-Verzeichnisse (odin)

  RAID-System mit 1 Terabyte, Quota pro Nutzer: 200 MB.

- Siehe: [http://www.informatik.uni-halle.de/studium/pools/]

Rechnernutzung, Software (2)

Rechnerpools (Forts.):

- Damit Sie auf den Rechnern im Pool 3.34 arbeiten können, muß ein Benutzerkonto eingerichtet werden. Die Daten entsprechen dem StudIP-Account.

  Hierzu melden Sie sich bitte vor der ersten Übung bei der Rechneraufsicht in Raum 333 (möglichst nach 14 Uhr). Bringen Sie Ihren Studentenausweis mit. Es wird das Start-Passwort von StudIP übernommen (das Sie mit Ihrer Immatrikulation bekommen haben).

- Remote Login (per `ssh`) z.B. auf Linux-Rechner **`anubis.informatik.uni-halle.de`**

  Sie können z.B. `PuTTY` für das remote Login nutzen.

Rechnernutzung, Software (3)

Rechnerpools (Forts.):

- Raum 335, 302: Uni-Pools (Anmeldung beim URZ)

Software auf Windows Rechnern:

- Microsoft Visual Studio 2010 Professional
- Eclipse, GNU C++ Compiler, Cygwin

Software auf Linux Rechnern:

- Eclipse, GNU C++ Compiler

Rechnernutzung, Software (4)

Falls Sie einen eigenen PC haben:

- GNU C++

  [http://gcc.gnu.org]. Auf Linux-Rechnern häufig schon vorinstalliert. Unter Windows: [http://www.mingw.org/] [http://www.cygwin.com/]

- Microsoft Visual C++, Express Edition (kostenlos)

  [http://www.microsoft.com/germany/express/products/windows.aspx]

- Borland (Borland C++ 5.5 ohne IDE kostenlos)

  [http://edn.embarcadero.com/article/20633]

- Eclipse (Java-basierte IDE) (Windows/Linux)

  [http://www.eclipse.org/], [http://www.eclipse.org/cdt/]

Rechnernutzung, Software (5)

Beispiele für kostenlose IDEs, Forts.:

- Code::Blocks (freie C++ IDE für Windows/Linux)
  [http://www.codeblocks.org/]
- KDevelop (freie IDE für Linux/UNIX)
  [http://www.kdevelop.org/]
- Sun Studio 12 (kostenlos) (für Solaris/Linux)
  [http://www.oracle.com/technetwork/server-storage/solarisstudio/]
- In der Wikipedia gibt es einen Vergleich integrierter Entwicklungsumgebungen.
  [http://en.wikipedia.org/wiki/
  Comparison_of_integrated_development_environments]
- Lesen Sie aber jeweils die Lizenzbedingungen!

Lehrbücher zu C++ (1)

- Ulla Kirch-Prinz, Peter Prinz:
  C++, Lernen und professionell anwenden.

  4. Aufl., mitp, 2007, ISBN 3-8266-1764-9, 928 Seiten, 44.95 €. Microsoft Visual C++ 2005 Express Edition auf beiliegender CD.

- Helmut Erlenkötter: C++, Objektorientiertes Programmieren von Anfang an.

  13. Aufl., Rowohlt, 2000, ISBN 3-499-60077-3, 332 Seiten, 10.95 €.

- Arnold Willemer:
  Einstieg in C++, 4. Auflage.

  Galileo Computing, 2009, ISBN 3836213850, 509 Seiten, mit CD-ROM, 24.90 €.

Lehrbücher zu C++ (2)

- Bjarne Stroustrup:
  The C++ Programming Language.

  Special Ed., Addison-Wesley, 2000, ISBN 0-201-70073-5, 1020 Seiten, 47.95 €.

  Deutsche Übersetzung: Die C++ Programmiersprache, 4. Auflage, Addison-Wesley, 2009, ISBN 3827328233, 44.00 €.

- Bjarne Stroustrup:
  Programming: Principles and Practice using C++.

  Addison-Wesley, 2008, ISBN 0-321-54372-6, 1272 Seiten, 41.95 €.

  Deutsche Übersetzung: Einführung in die Programmierung mit C++, Pearson, 2010, ISBN 3868940057, 69.95 €.

Lehrbücher zu C++ (3)

- Stanley B. Lippman, Barbara E. Moo, Josee Lajoie:
  C++ Primer, 4th Edition.

  Addison-Wesley, 2005, ISBN 0201721481, 912 pages, 36.95 €.

- Dirk Louis:
  C++: Programmieren mit einfachen Beispielen.

  Markt und Technik, 2009, ISBN 3827244838, 352 Seiten, 12.95 €.

- Heiko Kalista:
  C++ für Spieleprogrammierer, 3. Aufl.

  Hanser, 2009, ISBN 3446421408, 463 Seiten, 34.90 €.

Literatur für Fortgeschrittene

- Scott Meyers: Effective C++. 55 Specific Ways to Improve Your Programs and Designs.

  3rd Ed., Addison-Wesley, 2005, ISBN 0-321-33487-6, 297 Seiten, 33.89 Euro.

- Scott Meyers: More Effective C++. 35 New Ways to Improve Your Programs and Designs.

  Addison-Wesley, 1996, ISBN 0-201-63371-X, 318 Seiten, 35.89 Euro.

- Herb Sutter, Andrei Alexandrescu:
  C++ Coding Standards.

  Addison-Wesley, 2005, ISBN 0-321-11358-6, 235 pages, 25.95 €.

RRZN Handbücher

- RRZN: Programmiersprache C (Nachschlagewerk), 14. Aufl., ca. 3.70 Euro (Preis schwankt nach Liefermenge)

  Eine Auswahl der vom Regionalen Rechenzentrum Niedersachsen und seiner Kooperationspartner verfassten Schriften ist im Universitätsrechenzentrum, Kurt-Mothes-Straße 1, 06120 Halle, bei Frau Oelgarte (Raum 108, Tel. 55-21815) oder bei Frau Klotz (Raum 110) erhältlich. [http://www.urz.uni-halle.de/lehrgaenge/handbuecher/] Diese Bücher werden aber in der Übung verkauft (nicht im URZ).

- RRZN: C++ (Von C zu C++), 12. Aufl., ca. 3.60

  Weitere Lehrbücher aus der Reihe sind z.B.
  Windows XP — Grundlagen, 1. Aufl., 4.65 Euro
  Internet, 5. Aufl., 5.00 Euro.
  Java 2: Grundlagen und Einführung, 4. Aufl., 6.90 Euro.

Weitere Literatur (1)

- Goos, Zimmermann: Vorlesungen über Informatik 1 Grundlagen und funktionales Programmieren.

  4. Aufl., Springer, 2005, ISBN 3540244050, 400 Seiten, 26.95 €.

- Goos, Zimmermann: Vorlesungen über Informatik 2 Objektorientiertes Programmieren u. Algorithmen.

  4. Aufl., Springer, 2006, ISBN 3540244034, 375 Seiten, 26.95 €.

- Hans-Jürgen Appelrath, Jochen Ludewig:
  Skriptum Informatik. Eine konventionelle Einführung.

  5. Aufl., Teubner, 2000, ISBN: 3519421534, 462 Seiten, 27.90 €.

Weitere Literatur (2)

- Arnd Poetzsch-Heffter: Konzepte objektorientierter Programmierung (Mit einer Einführung in Java).

  Springer, 2009, ISBN 3540894705, 352 Seiten, 29,95 €.

- Brian W. Kernighan, Dennis M. Ritchie: The C Programming Language, 2nd Ed.

  Prentice Hall, 1988, ISBN 0-13-11-362-8, 272 Seiten, 38.45 Euro.

- INCITS/ISO/IEC 14882-2003: Programming Languages — C++

  Offizieller Standard. Erhältlich bei [http://webstore.ansi.org] für 30$.

Weitere Literatur (3)

- Daniel Molkentin: Qt4, Einführung in die Applikationsentwicklung.

  Open Source Press, 2006, ISBN 3937514120, 444 Seiten.
  2. Auflage (ISBN 3937514996) erscheint 2010, 44.90 €.

- Jasmin Blanchette, Mark Summerfield: C++ GUI Programming with Qt4, 2nd Ed.

  Prentice Hall, 2008, ISBN 0132354160, 752 Seiten, 43.95 €.

- [http://haweb1.bibliothek.uni-halle.de/]

Inhalt

Zeitliche Belastung (1)

- Für dieses Modul gibt es 5 Leistungspunkte (LP).

  Pro Semester soll man 30 LP erwerben, so dass man nach 6 Semestern den Bachelor-Grad (180 LP) erworben hat. Jeder LP entspricht einer durchschnittlichen Stundenbelastung von 25–30 Stunden, insgesamt also 125–150 Stunden (zum Teil in vorlesungsfreier Zeit zu leisten).

- Aufteilung von 150 Stunden Arbeitszeit:

Lernform	SWS	Stunden
Vorlesung	2	30
Praktische Übung	2	30
Hausaufgaben/direkte Nacharbeit	0	30
Selbststudium	0	45
Spezielle Prüfungsvorbereitung	0	15

Zeitliche Belastung (2)

- Die 150 Stunden sind nur ein Durchschnittswert, in den auch Studierende eingehen, die schon Programmierkenntnisse von der Schule mitbringen.
- Vorlesungen gehen deutlich schneller vorwärts als Schulunterricht: Es gibt nur wenig Wiederholung.

  Für die Wiederholung müssen Sie selbst sorgen, indem Sie sich zu Hause hinsetzen, die Folien und Ihre Notizen noch einmal durchgehen, darüber nachdenken, weitere Literatur (Bücher, Internet) studieren, die Hausaufgaben machen, und sich vielleicht noch eigene Programmierprojekte vornehmen. Das kann auch Spaß machen!

Zeitliche Belastung (3)

- Praktische Fähigkeiten wie die Programmierung erwirbt man neben dem
  - ◇ notwendigen theoretischen Wissen auch durch
  - ◇ praktisches Tun und Lernen aus Fehlern.
- Oft können auch Lösungen, die im Prinzip korrekt sind (funktionieren), noch verbessert werden.

  Z.B. einfacher, kürzer, übersichtlicher, leichter änderbar, resourcensparender (schneller/weniger CPU-Zeit oder weniger Speicherplatz). Nutzen Sie die Übung zum Austausch mit anderen Studierenden und dem Tutor (natürlich möglichst nach Abgabe der Hausaufgaben).
- Wie viel Sie lernen, hängt an der investierten Zeit.

Zeitliche Belastung (4)

- Wenn praktische Fähigkeiten auch notwendig sind, ist dies doch eine Vorlesung an einer Universität:
  - ◇ Man darf über C++ auch kritisch nachdenken.

    Mir gefällt auch nicht alles in C++. Überlegen Sie sich, was Sie als Programmiersprachen-Entwerfer anders machen würden.
  - ◇ Ziel ist auch ein möglichst leichter Übergang zu anderen Sprachen (allgemeine Konzepte!).
  - ◇ Auswendiglernen sollte nicht nötig sein.

    Wichtige Dinge prägen sich bei ausreichender theoretischer und praktischer Beschäftigung mit der Programmierung automatisch ein, selten benötigte Details kann man bei Bedarf nachschlagen.
  - ◇ Ihre Mitwirkung (Fragen, Vorschläge) ist wichtig!

Vorlesungs-Etikette

- Vermeiden Sie Verhalten, das Ihre Mitstudenten oder den Professor ablenkt:
  - ◇ Vermeiden Sie Gespräche während der Vorlesung.

    Glauben Sie nicht, dass Sie in der Masse untergehen: Der Professor kann durchaus sehen, wer sich unterhält. Wenn Sie Ihren Nachbarn etwas zur Vorlesung fragen müssen, machen Sie es leise und kurz. Wenn die Frage möglicherweise auch für andere interessant ist, stellen Sie sie offiziell (melden, ggf. rufen).
  - ◇ Wenn Sie zu spät kommen oder früher gehen müssen, setzen Sie sich möglichst an den Rand.
  - ◇ Notebooks sollten währen der Vorlesung nur die Folien anzeigen (eventuell Editor, Compiler).

Ratschläge zum Studium (1)

- Professoren freuen sich über Fragen!

  Selbst wenn der Professor die Frage nicht (gleich) beantworten kann, zeugt sie von Interesse, und am Ende lernen alle etwas davon.

- Lesen Sie möglichst mehrere Bücher zum Thema der Vorlesung, nicht nur das Skript.

  Seien Sie selbständig und etwas kritisch. Man kann Dinge auch ganz ohne Vorlesung lernen. Glauben Sie nicht etwas, nur weil Sie es zufällig gehört haben. Versuchen Sie zu verstehen, nicht auswendig zu lernen.

- Nehmen Sie das Studium ernst.

  Soviel Zeit, wie Sie jetzt haben, sich fortzubilden und Bücher zu lesen, haben Sie später nie wieder. Versuchen Sie, mit 6/10 Semestern auszukommen, wenn es keine besonderen Gründe gibt.

Ratschläge zum Studium (2)

- Lernen Sie programmieren (und wirklich gut).
  Es ist das Handwerk, auf dem alles beruht.
- Lernen Sie mathematisches Denken (Formalisieren, Beweisen), und Techniken wie z.B. vollständige Induktion, Grundlagen der Algebra und Logik.
- Arbeiten Sie in Gruppen zusammen, aber sorgen Sie dafür, dass jedes Gruppenmitglied alles lernt.
- Computerspiele können süchtig machen.
- Falls Frage nach Sinn des Lebens: Ich bin Christ.