



**Sommersemester 2006: Termine zur Vorlesung
„Grundgebiete der Informatik 2:
Algorithmen und Programmiertechniken“**

Veranstaltung	Wochentag	Zeit	Ort	Beginn	Person
Vorlesung	Freitag	10:00–11:30	AM	07.04.	Prof. Nagl
Übungsgruppe	Montag	15:30–16:15	Be 114	10.04.	Jonathan Heinen
		16:30–17:15			
	Dienstag	14:00–14:45	F60	11.04.	Nils Jansen
		15:00–15:45			
	Mittwoch	12:15–13:00	RS 4	12.04.	Thorsten Fuhrmann
		13:15–14:00			
	Mittwoch	12:15–13:00	RS 5	12.04.	Tim Hemig
		13:15–14:00			
	Donnerstag	12:15–13:00	SG 413	13.04.	Ingo Esser
		13:15–14:00			
C++-Betreuung	Montag	11:30–13:15	CIP-Pool	10.04.	wechselnd
	Donnerstag	8:30–10:30		13.04.	
Fragestunde	Freitag	14:30–16:00	AH V	21.04.	Assistenten

Die Lage der Übungsräume kann dem Anhang des Vorlesungsverzeichnisses entnommen werden. In der Exkursionswoche vom 05.06.2006 bis zum 09.06.2006 finden keine Übungen statt. Die Übungsaufgaben der Vorwoche sind deshalb umfangreicher. Der CIP-Pool des Rogowski-Instituts für Elektrotechnik befindet sich im 2. Stock des Seminargebäudes. Dort stehen zu den angegebenen Zeiten Ansprechpartner zur C++-Programmierung zur Verfügung.

Die Anmeldung zu den Übungsgruppen kann am Tag der ersten Vorlesung auf der Webseite zu dieser Veranstaltung (<http://www-i3.informatik.rwth-aachen.de/ggdi>) unter der Rubrik Übungsgruppen vorgenommen werden.

Die Klausuren finden am 20. 07. 2006 und am 05. 09. 2006 statt. Genauer wird auf unserer Webseite bekanntgegeben.

Während des Semesters wird eine Probeklausur voraussichtlich am 16. 06. 2006 um 16:30 Uhr angeboten, die Teilnahme ist empfehlenswert! Als weitere Hilfe zur Klausur veranstalten wir zwei Vorbereitungstermine, die Daten werden noch bekanntgegeben.



Nagl, Becker, Ranger, Wörzberger

Übungen zur Vorlesung „Grundgebiete der Informatik 2: Algorithmen und Programmiertechniken“

— Blatt 1 —

Allgemeines zu den Übungen

Die Aufgabenblätter werden immer freitags am Ende der Vorlesung verteilt. Die bearbeiteten Übungen müssen jeweils am darauffolgenden Freitag *vor* der Vorlesung abgegeben bzw. bis zum gleichen Zeitpunkt per E-Mail an `ggdi2-xxx@i3.informatik.rwth-aachen.de` geschickt werden (für xxx jeweils den Vornamen des Übungsgruppenleiters einsetzen, also z.B. `ggdi2-tim@i3.inf...`). Bitte schreiben Sie Ihren Namen und den Übungstermin bzw. den Namen Ihres Übungsleiters auf Ihre Lösungen, damit wir sie richtig zuordnen können. Korrigierte Übungsblätter werden in den Übungsgruppen der nachfolgenden Woche zurückgegeben. Die Übungsgruppen finden ab Kalenderwoche 15 statt.

Zur Bearbeitung der Übungen müssen Arbeitsgruppen von drei Teilnehmern gebildet werden. Die Übungstermine sollten für Fragen aller Art genutzt werden.

Für einige Aufgaben ist ein Programm in der Programmiersprache C++ zu erstellen. Als Lösung ist ein syntaktisch korrektes Programmlisting abzugeben *und* dem Übungsgruppenleiter elektronisch zu übermitteln (per E-Mail). Dieses Programm ist mit einem Compiler zu überprüfen. Dazu können eigene Compiler oder der des CIP-Pools verwendet werden. Dort kann während der normalen Öffnungszeiten programmiert werden, zusätzlich bieten wir jeweils Montags zwischen 11:30 und 13:15 sowie Donnerstags zwischen 8:30 und 10:30 Uhr eine Beratung an. Bitte stellen Sie Fragen zu C++ oder zu den Aufgaben **nicht** dem Beratungspersonal des CIP-Pools, sondern nur den Betreuern der Übungsgruppen.

Verweise auf frei verfügbare C++-Compiler finden sich (neben anderen Informationen zur Vorlesung) auf <http://www-i3.informatik.rwth-aachen.de/ggdi>.

Um Ihren Lernerfolg zu überprüfen, bieten wir Ihnen an, zu dieser Veranstaltung einen Übungsschein zu erwerben. Jeder Aufgabe ist eine Punktzahl zugeordnet. Um sich für den Schein zu qualifizieren, müssen Sie über das Semester hinweg mindestens 50% der Gesamtpunktzahl aller Aufgaben erreichen. Zusätzlich müssen Sie mindestens eine Aufgabe in Ihrer Übungsgruppe vorrechnen. Ihr Übungsgruppenleiter wählt für jede Aufgabe unangekündigt einen Teilnehmer zum Vorrechnen aus. Nur wer dann anwesend ist und vorrechnet, hat ein Anrecht auf den Schein. Für Studierende, die keinen Schein erwerben möchten, ist die Teilnahme an der Übung und das Vorrechnen freiwillig.

Unter `news://news.rwth-aachen.de/rwth.elektrotechnik.ai2` ist eine Newsgroup zur Vorlesung eingerichtet. Hier können Sie untereinander Erfahrungen austauschen.

1. Aufgabe Aufbau von EBNF-Regeln:

(13 Punkte)

In der Vorlesung haben Sie zur Syntaxbeschreibung (Grammatik) von Programmiersprachen EBNF-Regeln kennengelernt. EBNF-Regeln bestehen aus Terminalsymbolen und Nichtterminalsymbolen, die mit bestimmten Operatoren verknüpft werden. Terminalsymbole sind Elemente der beschriebenen Sprache, beispielsweise Schlüsselwörter wie `class` oder `if` in C++. Nichtterminalsymbole definieren einerseits komplexe Spracheinheiten, andererseits helfen sie die Syntaxbeschreibung zu strukturieren.

- (a) Als Konstruktionsmöglichkeiten in EBNF-Regeln gibt es Sequenz, Option, Wiederholung, Alternative und Rekursion. Geben Sie für jede dieser Konstruktionsarten drei Regeln (beziehungsweise zwei rekursiv definierte Nichtterminale für die Rekursion) aus der C++-Syntax im Anhang I des Skriptes an, in denen sie vorkommen. (3 Punkte)

- (b) Gegeben ist folgende vereinfachte Grammatik für if-Statements in C++.

$ifStatement ::= \text{if (expression) \{ statementList \}}$
 $\quad \quad \quad \{ \text{else if (expression) \{ statementList \}} \}$
 $\quad \quad \quad [\text{else \{ statementList \}}]$ (1)

$expression ::= simpleExpression [relation simpleExpression]$ (2)

$relation ::= == \mid != \mid < \mid <= \mid > \mid >=$ (3)

$simpleExpression ::= [+ \mid -] factor \{ addOperator factor \}$ (4)

$factor ::= number \mid ident \mid (expression) \mid !factor$ (5)

$statementList ::= statement \{ statement \}$ (6)

$addOperator ::= + \mid - \mid \parallel \mid \&\&$ (7)

$statement ::= ifStatement \mid \text{cout} << string ;$ (8)

Hier soll *number* eine beliebige Ziffernfolge, *string* eine in Anführungszeichen eingeschlossene beliebige Zeichenfolge und *ident* eine Buchstabenfolge (Identifizierer) sein.

Leiten Sie die folgenden Statements aus *ifStatement* ab.

(4 Punkte)

- `if (spielEnde) { cout << "Fertig"; }`
- `if (s+k == 21) {
 cout << "Gewonnen!";
} else if (s+k < 21) {
 cout << "Weiter";
} else {
 cout << "Verloren";
}`

- (c) Verändern und erweitern Sie die Grammatik aus (b) so, daß in Ausdrücken auch Multiplikation und Division verwendet werden kann. Sie können dazu ein neues Nichtterminal *term* einführen und in Regel (4) statt *factor* verwenden. Stellen Sie sicher Sie, daß ihre erweiterte Grammatik „Punkt- vor Strichrechnung“ beachtet.

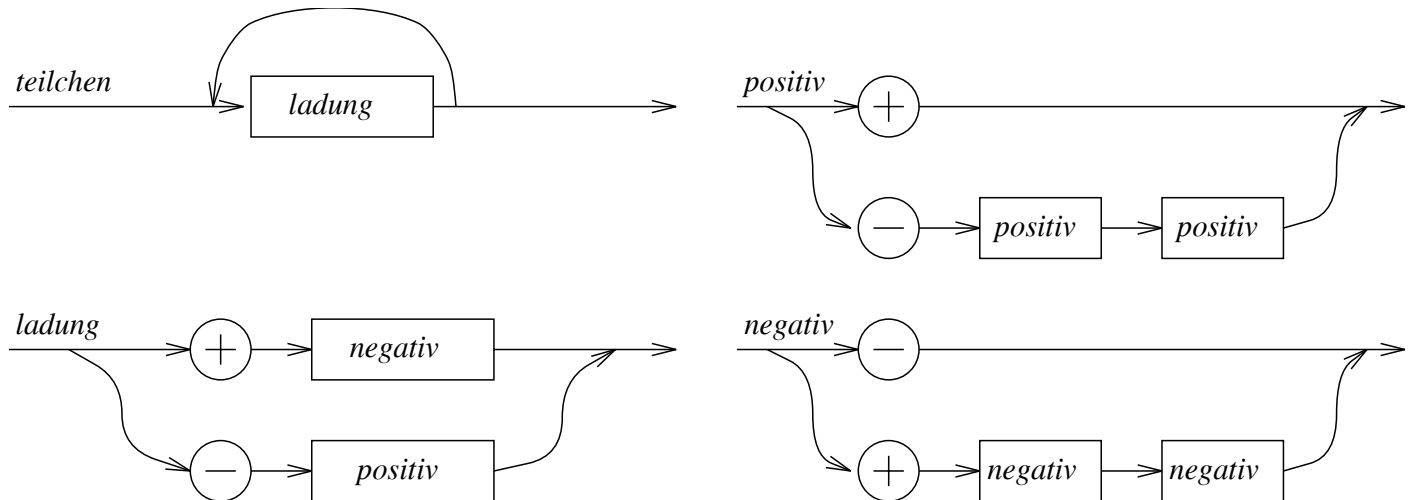
(6 Punkte)

2. Aufgabe Syntaxdiagramme:

(11 Punkte)

Statt EBNF-Regeln kann man für eine Syntaxbeschreibung auch Syntaxdiagramme verwenden. Syntaxdiagramme sind insbesondere für Ungeübte leichter zu verstehen, sind dafür andererseits weniger kompakt als EBNF-Regeln.

Gegeben seien die folgenden Syntaxdiagramme:



- (a) Man kann EBNF-Regeln systematisch in Syntaxdiagramme übersetzen. Geben Sie dazu für Sequenz, Wiederholung, Option und Alternative Syntaxdiagramme an, die als Basisbausteine bei der Übersetzung dienen können. Warum haben wir die Rekursion ausgelassen? (3 Punkte)
- (b) Übersetzen Sie die Diagramme in EBNF. (4 Punkte)
- (c) Finden Sie eine Grammatik (in EBNF oder Syntaxdiagrammen), mit der die gleichen Worte erzeugt werden, die aber weniger Regeln hat. (4 Punkte)

3. Aufgabe Mein erstes Programm:

(15 Punkte)

Machen Sie sich mit Editor und Compiler vertraut! Frischen Sie ihre Kenntnisse aus den Versuchen 4–8 aus dem Praktikum “Informatik I” anhand der folgenden Aufgaben auf:

- (a) Schreiben Sie ein Programm, das die Namen der Mitglieder Ihrer Gruppe ausgibt. Übersetzen und starten Sie es. (Dieser Quelltext braucht nicht abgegeben zu werden.)
- (b) Schreiben Sie ein Programm, das Zahlen von der Standardeingabe liest, bis diese endet und die Gesamtsumme ausgibt. Das Programm braucht bezüglich der Eingabe nicht besonders robust zu sein. (Dieser Quelltext braucht nicht abgegeben zu werden.)
- (c) Teilen Sie Ihr Programm aus (b) in mehrere Quelltexte auf: `main.cc` (Leseschleife), `summe.cc` (Aufsummieren und Abfragen) und `summe.h` (Deklaration der Funktionen in `summe.cc`). (6 Punkte)
- (d) Lesen Sie in der Dokumentation und schreiben Sie ein Programm unter Verwendung der Funktionen `time`, `localtime` und `asctime`, das die Sekunden seit Anfang 1970 sowie die aktuelle Zeit in der vor-eingestellten Zeitzone anzeigt. (9 Punkte)