Inhaltsverzeichnis

1 Übung: Editor, Arbeiten mit der Kommandozeile 2

2 Übung: Wiederholung, E-Mail, WWW 5

3 Übung: Zugriffsrechte, Jokerzeichen, etc. 8

4 Übung: Umleitung, Prozesse, Algorithmen 11

5 Übung: Einführung in die Programmierung mit `Modula-3` 13

6 Übung: Numerische und ordinale Typen 16

7 Übung: Auswahlstrukturen und Sichtbarkeit 18

8 Übung: Unterprogramme 21

9 Übung: Schleifen 23

10 Übung: Felder und Texte 24

11 Übung: Datenverbünde, Zeiger, Funktionsvariablen 26

12 Übung: Umgang mit Dateien und Fehlerbehandlung 28

13 Übung: Module und Objekte 29

14 Übung: Zusatzaufgaben 30

15 Übung: Aufgaben auf Halde 40

1 Übung: Editor, Arbeiten mit der Kommandozeile

Aufgabe 1 – Der Editor `nedit`

Rufen Sie den Editor `nedit` auf und tippen Sie den folgenden Text ein (mit Tippfehlern!):

```
Der xxx ist ein seer leistungspfaehiger Konditor, mit dem
man sehr affektiv Texte bearbeiten kan. Es gipt diferse Zusatzpakete,
die xxx an die Betuerfnisse der jeweiligen Tateidypen anpazen.

Wichtige Befehle von xxx:
 ctrl-s  speichern
 ctrl-f  suchen
 ctrl-r  suchen und ersetzen
 ctrl-j  Abschnitt formatieren
 ctrl-z  Änderung zurücknehmen
```

Speichern Sie den Text unter dem Namen `editor.txt` in Ihrem Heimverzeichnis ab. Verlassen Sie den Editor. Rufen Sie den Editor erneut auf, öffnen Sie die Datei `editor.txt` (per `File`-Menü und `Open`) und korrigieren Sie die Tippfehler.

Kopieren Sie dann mit `Copy` und `Paste` (`Edit`-Menü) den Anfang an das Ende des Textes und verwenden Sie `Replace...` (im `Search`-Menü), um `'xxx'` durch `'nedit'` zu ersetzen. Der Text sollte nun so aussehen:

```
Der nedit ist ein sehr leistungsfaehiger Editor, mit dem
man sehr effektiv Texte bearbeiten kann. Es gibt diverse
Zusatzpakete, die nedit an die Beduerfnisse der jeweiligen
Dateitypen anpassen.

Wichtige Befehle von nedit:
 ctrl-s  speichern
 ctrl-f  suchen
 ctrl-r  suchen und ersetzen
 ctrl-j  Abschnitt formatieren
 ctrl-z  Änderung zurücknehmen

Der nedit ist ein sehr leistungsfaehiger Editor!
```

Speichern Sie den korrigierten Text unter dem neuen Namen `editor_korrigiert.txt` ab. Wenn Sie wollen, können Sie den Text ausdrucken.

Aufgabe 2 – Drucken

a) Starten Sie `mozilla` oder `netscape` und lesen Sie auf der Seite

`http://www.informatik.uni-bremen.de/t/info/drucken.html`

die Dokumentation zum Drucksystem.

b) Lassen Sie sich Ihr Druckkontingent anzeigen (`pacc`).

c) Drucken Sie eine Datei aus und betrachten Sie sich die Druckerschlange (mit `lpq` bzw. `lpq -l`). Falls Ihr Auftrag erscheint, löschen Sie diesen.

d) Falls die Datei noch nicht gedruckt worden ist, holen Sie dies nun nach.

e) Kopieren Sie die Datei `einfuehrungUNIX.ps` aus dem Verzeichnis
`/home/zetem/daten/ausbildung/rechprakt`
in Ihr Heimverzeichnis.

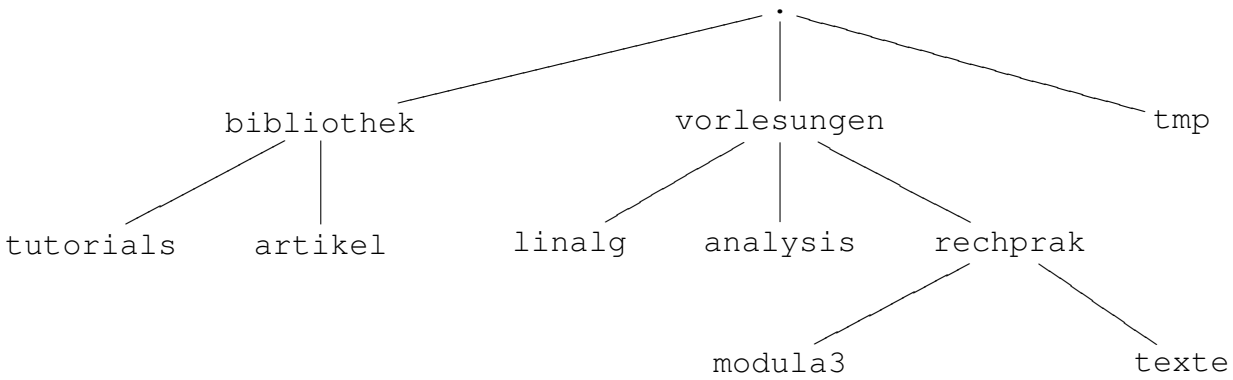
f) Wandeln Sie die kopierte Datei in ein dreispaltiges Dokument um:
`psnup -n 3 einfuehrungUNIX.ps >einfuehrungUNIX_3up.ps`

g) Drucken Sie die Datei mit dem Befehl `lpr einfuehrungUNIX_3up.ps` aus.

h) Lassen Sie sich erneut Ihr Druckkontingent anzeigen.

Aufgabe 3 – Umgang mit Verzeichnissen

Erzeugen Sie folgende Verzeichnisstruktur in Ihrem Heimverzeichnis:



a) Wie können diese Verzeichnisse alle auf einmal erzeugt werden?

b) Verschieben Sie die Datei `editor.txt` aus Aufgabe 1 in das Verzeichnis `texte`.

c) Gehen Sie in die oberste Ebene Ihres Heimverzeichnisses und starten Sie dort den Editor `nedit`. Erzeugen Sie eine Datei mit dem Namen `test.txt` und speichern Sie diese direkt im Verzeichnis `texte` ab.

d) Lassen Sie sich den Inhalt von `texte` ausgeben (ohne in das Verzeichnis zu wechseln).

e) Kopieren Sie beide Dateien in das Verzeichnis `tmp`.

f) Löschen Sie den kompletten Zweig `vorlesungen` und verwenden Sie hierfür einen Befehl, der Sie jede Aktion bestätigen lässt.

g) Erstellen Sie das Verzeichnis `vorlesungen` erneut samt Unterverzeichnissen und verschieben Sie die beiden Dateien aus dem Verzeichnis `tmp` wieder in das Verzeichnis `texte`.

h) Wieviele Dateien existieren in Ihren Heimverzeichnis? Schauen Sie sich dazu den Inhalt mit `ls`, dann mit `ll` und dann noch mit der Option `-a` an.
(Löschen Sie keine der versteckten Dateien `.datei`!)

2 Übung: Wiederholung, E-Mail, WWW

Aufgabe 4 – Noch einmal drucken

a) Kopieren Sie alle Dateien aus dem Verzeichnis
`/home/zetem/daten/ausbildung/rechprakt/tutorial`
in Ihr Verzeichnis `tutorial`.

b) Entpacken Sie die Datei `erste-schritte.ps.gz` mit dem Befehl `gunzip`. Wie groß ist die Datei? Danach verpacken Sie sie wieder mit `gzip`. Um wieviel kleiner ist die Datei nun?

c) Schauen Sie sich diese Postscript-Datei mit dem Programm `gv` an, `gv` entpackt sie dazu automatisch an einem ruhigen Plätzchen.

d) Falls Sie die Datei ausdrucken wollen, verwenden Sie der Kommandozeile
`psbook erste-schritte.ps | lprx -2 -X`

e) Lassen Sie sich Informationen zu den beiden verwendeten Befehlen anzeigen und suchen Sie sich heraus, was die verwendeten Parameter bedeuten. Benutzen Sie `man befehl` und `apropos befehl` für orginale UNIX-Befehle. Für Nicht-Standard-Befehle bekommt man oft durch `befehl -h`, `befehl -?` oder `befehl --help` Informationen angezeigt.

f) Lassen Sie sich erneut Ihr Druckkontingent anzeigen.

Mit dem Befehl `lprx` lassen sich auch mehrere Dateien auf einmal ausdrucken. Dies ist bei Verwendung der Option `-2` besonders platzsparend. Zur Erinnerung: der Befehl `lprx` ist kein Standardbefehl, der Standarddruckbefehl lautet `lpr`. Zum platzsparenden Ausdrucken gibt es außerdem den Befehl `a2ps`. Das Programm `a2ps` bereitet auch reine Texte und Programmtexte nett auf.

Aufgabe 5 – Noch einmal Verzeichnisse und `nedit`

a) Erzeugen Sie einen weiteren Verzeichnisbaum in Ihrem Heimverzeichnis. Das oberste Verzeichnis soll `privat` genannt werden. Die Baumstruktur und die Namen der Unterverzeichnisse sollten Ihren Bedürfnissen entsprechen. Ein Unterverzeichnis soll den Namen `briefe` tragen.

b) Schreiben Sie mit `nedit` zwei Dateien (Briefe) und speichern Sie diese in `briefe`.

c) Legen Sie ein Verzeichnis `sicherungskopien` an und kopieren Sie dorthinein alle bisher von Ihnen erstellten Texte. Erzeugen Sie eine weitere Kopie von jeder Datei, wobei die Dateinamen erweitert werden sollten zu der Form `datei.txt.kopie`. (Mit dem bisher erlangten Wissen können Sie diesen Vorgang vermutlich nicht automatisieren.) Jede Datei existiert nun zweimal.

d) Löschen Sie die Dateien in `sicherungskopien`, die nicht auf `.kopie` enden.

e) Zu den in Aufgabe 4 kopierten Dateien gehört auch `ref-emacs.ps`, die Sie sich wieder mit `gv` anschauen und, wenn gewünscht, ausdrucken können.

Aufgabe 6 – Email mit WebMail

Machen Sie sich mit dem WWW-basierten E-Mail-Service der Universität Bremen unter

`https://webmail.informatik.uni-bremen.de/`

vertraut.

a) Steuern Sie die entsprechende Seite mit ihrem Brauser an, nicken Sie die Sicherheitshinweise ab und loggen Sie sich mit ihrem UNIX-Benutzernamen und Passwort ein.

b) Schicken Sie Ihrem Nachbarn eine Email und sich selbst eine Kopie der Email.

c) Beantworten Sie die erhaltene Email.

d) Erstellen Sie sich eine Signatur für Ihre E-Mails, also einen Text, der unter jeder versandten E-Mail steht.

e) Erstellen Sie eine Email mit attachment (z.B. eine der erstellten Textdateien) und senden Sie diese an ihren Nachbarn.

f) Speichern Sie von der erhaltenen Email das attachment ab.

g) Erstellen Sie einen Ordner (folder) für die Emails des Rechnerpraktikums und legen Sie alle erhaltenen Emails dort ab.

Aufgabe 7 – Abonnieren von Email-Verteilern

Damit alle Studenten rechtzeitig von studentischen Veranstaltungen erfahren, gibt es einen Email-Verteiler. Bei diesem meldet man sich an, indem man eine E-Mail an

majordomo@majordomo.zfn.uni-bremen.de

mit dem Text

```
subscribe stud-math
```

schickt. Als Absender muss man die Email-Adresse eintragen, an welche man die Ankündigungen geschickt haben möchte.

Nach dem man diese Mail gesendet hat, schickt einem das Verteilersystem Informationen über die Bedienung zu.

Aufgabe 8 – WWW – World Wide Wait

a) Finden Sie auf dem Bremer-Uni-Lageplan den Raum GW 2 A4050 (Medienstelle der Uni Bremen).

b) Suchen Sie Bücher über UNIX in der SuUB heraus.

c) Wählen Sie ein Buch aus und versuchen Sie herauszufinden, wie man dieses über Internet bestellen kann.

d) Suchen Sie mit einer Suchmaschine (Google o.ä.) Tutorials zum Thema `Modula-3`-Programmierung und speichern Sie gefundene Tutorials in Ihrem Verzeichnis ab.

e) Haben Sie sich schon einmal über den StugA informiert?

f) Und was macht eigentlich das ZeTeM?

g) Viele Infos zur Mathematik in Deutschland bietet `http://www.mathematik.de/`

h) Wie wird man als Mathematiker zum Millionär:
`http://www.claymath.org/millennium/`

i) Finden Sie Näheres über berühmte Mathematikerinnen heraus, z.B. auf den Seiten `http://www-history.mcs.st-and.ac.uk/`

j) Suchen Sie eine Zugverbindung für Ihren Ausflug am nächsten Wochenende nach Berlin heraus (mit Preiskalkulation), z.B. mit `http://www.bahn.de/`

k) Schauen Sie unter `http://www.bremen.de/` nach, was Bremen heute Abend zu bieten hat.

l) ....

3 Übung: Zugriffsrechte, Jokerzeichen, etc.

Aufgabe 9 – Zugriffsrechte

Angenommen, Sie wollen einem Ihrer Mitstudenten eine Datei übermitteln und umgekehrt Dateien von Ihren Mitstudenten erhalten. Dafür ist es sinnvoll, die Zugriffsrechte des `tmp`-Verzeichnisses so zu ändern, dass auch andere dort lesen und schreiben können.

a) Verändern Sie die Rechte Ihres `tmp`-Verzeichnisses entsprechend.

b) Legen Sie mit `nedit` eine Datei für Ihren Nachbarn an.

c) Kopieren Sie diese Datei in das `tmp`-Verzeichnis Ihres Nachbarn.

d) Nun besteht das Problem, dass die von Ihnen kopierte Datei Ihnen (!) und nicht Ihrem Nachbarn gehört. Er kann sie deshalb nicht verändern. Verändern sie also unbedingt noch die Zugriffsrechte der kopierten Datei so, dass diese auch von Ihrem Nachbarn geändert werden kann.

e) Schließlich betrachten Sie die hoffentlich von Ihrem Nachbarn in Ihrem Verzeichnis abgelegte Datei mit dem `more`-Befehl.

Dieses Verfahren ist sinnvoll, wenn man große Dateien übermitteln will. Große Dateien per Email zu versenden, ist unnötig aufwändig, dauert lange und wird oft auch nicht zugelassen.

Aufgabe 10 – Wildcards und Umleitungen

Nehmen wir an, Sie tippen ein großes Dokument wie etwa ein Buch. Aus logischer Sicht ist dies in viele kleine Teile unterteilt, wie z.B. Kapitel und Abschnitte. Physikalisch sollte es auch so unterteilt sein, denn große Dateien zu editieren ist mühsam. Angenommen Sie besitzen folgende Dateien, wovon einige temporäre Dateien nur als Zwischenspeicher dienen, z.B. für mehrfach verwendeten Text:

```
ch1.1      ch2.2      tempB      tmp1
ch1.2      ...        tempb      tmp11
ch1.3      ch9.1      tempk
...        ch9.2      temps
ch2.1      ch9.3      tempt
```

Hinweis: Als ordentlicher Autor verwenden Sie natürlich keine Dateinamen mit Kapitelnummern, die kann ein System wie LaTeX besser vergeben. Sinnvoll sind Dateinamen, die den Inhalt eines Abschnittes widerspiegeln.

Überlegen Sie sich, was folgende Kommandos bewirken!

```
-> ls ch[12346789]*
-> ls ch[1-46-9]*
-> rm temp[a-z]
-> ls ?
-> ls ch?.1
-> rm tmp?
```

Aufgabe 11 – Suchpfad und History

a) Wie lautet Ihr Suchpfad? Geben Sie Ihren Suchpfad in eine Datei aus.

b) Erzeugen Sie ein Unterverzeichnis `programme` in dem Verzeichnis `rechprak`.

c) Ändern Sie den Suchpfad so, dass Befehle zuerst im Unterverzeichnis `programme` Ihres Heimverzeichnisses gesucht werden.

d) Lassen Sie sich mit dem Befehl `history` die Kommandos anzeigen, die sie zuletzt eingegeben haben.

Aufgabe 12 – `alias`

Um neue Befehle zu definieren, kann der Befehl `alias` verwendet werden, z.B.:

`alias p='pwd'` erstellt eine Abkürzung von `pwd`

`alias ls='ls -al'` ersetzt den Befehl `ls` durch den Befehl `ls -al`

Die so erstellten, neuen Befehle sind nur in der aktuellen Shell gültig.

Aufgabe 13 – Die Datei `.private_functions`

Wird eine Pfaderweiterung wie in Aufgabe 11 erstellt, so ist diese nur in der aktuellen Shell gültig. Das gleiche gilt für die neu gebildeten Befehle aus Aufgabe 12.

Sollen die Änderungen für jede neue Shell zur Verfügung gestellt werden, so sollte man sie in die Datei `.privat_functions` eintragen. Jede neue Shell arbeitet dann zuerst diese Befehle ab. Wir wollen die Funktionsweise dieser Datei zunächst mit dem `echo`-Befehl testen und dann die neuen Befehle aus Aufgabe 12 eintragen.

a) Wechseln Sie in die oberste Ebene Ihres Heimverzeichnisses.

b) Öffnen bzw. erzeugen Sie mit `nedit` die Datei `.privat_functions`.

c) Tragen Sie hier zur Begrüßung einen weiteren `echo`-Befehl ein, speichern Sie die Datei und testen Sie den Eintrag, indem Sie eine neue Shell öffnen.

d) Tragen Sie nun die neuen Befehle aus Aufgabe 12 in diese Datei ein und testen Sie die Einträge.

e) Sie können auch das primäre Promptzeichen ändern, z.B. durch den Eintrag von

```
export PS1='\u@\h:\w>'
```

Bitte gehen Sie mit Änderungen von versteckten Dateien äußerst vorsichtig um!

4 Übung: Umleitung, Prozesse, Algorithmen

Aufgabe 14 – Umleitungen

Erklären Sie, warum bei

```
-> ls > ls.out
```

`ls.out` in der Liste der Dateinamen auftaucht.

Testen Sie diesen Befehl und betrachten Sie sich anschließend die entstandene Datei. Schließlich sollte diese wieder gelöscht werden.

Was passiert bei

```
-> cat x.dat y.dat > y.dat
```

und bei

```
-> cat x.txt >> x.txt   ?
```

Durch welche Verknüpfung von Kommandos in einer Kommandozeile kann man den Inhalt von `x.txt` „verdoppeln“?

Aufgabe 15 – Zeichenketten suchen

Wie oft kommt die Zeichenkette „xx“ in der Datei `editor.txt` aus Aufgabe 1 vor? Suchen Sie dazu diese Zeichenkette mit dem Kommando `grep`. Wie oft kommt die Zeichenkette „be“ vor? Benutzen Sie dazu einmal das Kommando `grep` und einmal `grep -i`. Was unterscheidet diese beiden Befehle?

Aufgabe 16 – Prozesse

a) Starten Sie `gv` im Hintergrund und `nedit` im Vordergrund. Stoppen Sie den Vordergrundprozess und lassen Sie sich alle laufenden Programme anzeigen. Stellen Sie den Vordergrundprozess in den Hintergrund und lassen sie sich wieder alle Programme anzeigen.

b) Lassen Sie sich alle laufenden Prozesse samt PID anzeigen. Brechen Sie die beiden von Ihnen gestarteten Prozesse ab. Testen Sie den Befehl `ps -e`. Worin liegt der Unterschied zu `ps -u nutzer`?

c) Testen Sie das Kommando `top` (Eingabe von `q` beendet das Kommando, Eingabe von `k` und der PID eines Prozesses bricht diesen ab).

Aufgabe 17 – Securelogin

Loggen Sie sich von Ihrer Maschine auf der Ihres Nachbarn ein. Lassen Sie sich anzeigen, wer gerade auf Ihrem Rechner eingeloggt ist.

Zum Einloggen und Kopieren zwischen verschiedenen Rechnern immer nur sichere Dienste wie `ssh`, `slogin`, `scp`, `sftp` verwenden!

Aufgabe 18 – Algorithmische Formulierungen

Unter Verwendung der drei algorithmischen Sprachelemente

Sequenz:	`";"`
Negation:	`NICHT( )`

Bedingte Anweisung:	`FALLS ...`
	`DANN ...`
	`SONST ...`

Wiederholung:	`SOLANGE ...` `FÜHRE AUS`

und den elementaren Äktionen und Bedingungen:

a) schlage erste Seite auf

b) lies ersten Namen der Seite

c) lies nächsten Namen

d) umblättern

e) letzter Name auf Seite?

f) Name gefunden?

versuche man einen Algorithmus zu formulieren, mit dem man einen Namen in einem Telefonbuch findet.

Aufgabe 19 – Wiegen

Wir verfügen über eine bestimmte Anzahl von Münzen, die identisch aussehen und die alle gleich schwer sind, mit Ausnahme von einer Münze, die schwerer ist als die anderen. Weiterhin verfügen wir über eine Balkenwaage, mit der bestimmt werden kann, ob eine Gruppe von Münzen (bestehend aus einer oder mehreren Münzen) schwerer ist als eine andere Gruppe.

Formulieren Sie einen Algorithmus (unter Verwendung der algorithmischen Sprachelemente aus Aufgabe 18), der feststellt, welches die schwerere Münze ist. Welche elementaren Aktionen und Bedingungen haben Sie dabei benutzt?

5 Übung: Einführung in die Programmierung mit `Modula-3`

Zunächst müssen wir für diese und alle folgenden Sitzungen der Shell erklären, wo die Programme für die `Modula-3`-Entwicklung installiert sind. Dies soll sowohl unter Linux als auch Solaris funktionieren. Kopieren Sie den folgenden Text aus der Datei

`/home/zetem/daten/ausbildung/rechprakt/setm3paths`

in ihre Datei `.private_functions`:

```
if [ `uname -s` = "Linux" ]; then
  export M3TARGET=LINUXLIBC6
fi
if [ `uname -s` = "SunOS" ]; then
  export LD_LIBRARY_PATH=/home/sparc-solaris8/lib:$LD_LIBRARY_PATH
  export M3TARGET=SOLgnu
fi
export PATH=/home/cm3/bin/$M3TARGET:/home/m3/$M3TARGET/bin:$PATH
```

Wer mit `nedit` arbeitet, kann von Syntax-Hervorhebung und automatischer Programmtextformattierung (durch das Programm `m3pp`) profitieren, wenn er `nedit` mit

```
nedit -import /home/zetem/daten/ausbildung/rechprakt/m3-nedit-mode
```

aufruft, dann im Menü `Preferences / Default Settings / Syntax Highlighting / On` aktiviert und dann diese neuen Einstellungen mit dem Menüpunkt `Preferences / Save Defaults...` dauerhaft speichert. Für den aktuell gestarteten `nedit` muss man noch `Preferences / Highlight Syntax` anwählen.

Die neuen Einstellungen für `nedit` bezüglich `Modula-3` bewirken folgendes:

- Schlüsselworte, Kommentare usw. werden hervorgehoben.
- Nach der Eingabe einer öffnenden Klammer wird zusätzlich die entsprechende schließende Klammer angehängt.
- Nach dem Beenden der Kopfzeile einer Programmstruktur wie **`IF THEN`** wird die folgende Zeile eingerückt.
- Über den Menüpunkt `Shell / indent` kann man den Programmtext übersichtlich formatieren, sofern das Programm syntaktisch korrekt ist.

Um leichter ein neues Programm schreiben zu können, legt das Programm `createm3skel` (genaugenommen ein Shell-Skript) die nötigen Verzeichnisse und eine Datei für das Hauptprogramm an:

```
createm3skel aufgabe21.1
```

erzeugt im aktuellen Verzeichnis ein neues Projekt mit Verzeichnis `aufgabe21.1`.

Aufgabe 20 – Variablen- und Funktionsnamen

Welche der folgenden Bezeichner sind in `Modula-3` zulässig?

a) `x`
b) `feb.mar`
c) `x1x`
d) `john brown`
e) `re-active`
f) `x(3)`
g) `feb24`
h) `24feb`
i) `A+B`
j) `mat_mul`
k) `2x`
l) `_feb_mar`

Aufgabe 21 – Formatierte Zahlenausgabe

Schreiben Sie ein Programm, das die Zahlen 1, 12, 123, -1, -12, -123 in verschiedenen Formaten ausgibt. Das Programm soll zunächst die Aufgabennummer eingerahmt ausgeben, dann die jeweilige Aufgabenstellung und schließlich die so formatierten Zahlen:

a) Ausgabe der Zahlen direkt hintereinander.
b) Ausgabe mit jeweils einem Zeichen Abstand.
c) Ausgabe untereinander und rechtsbündig. Für jede Zahl sollen 10 Stellen reserviert werden und sie soll mit „x“ vorne und hinten eingerahmt werden.
d) Wie c), nur Leerstellen mit 0 aufgefüllt.
e) Wie in c), nur linksbündig.

Aufgabe 22 – Rechenoperationen

a) Ausdrücke in `Modula-3`:
   1) Welchen Wert ergibt `17 DIV 3`?
   2) Welchen Wert ergibt `17 MOD 3`?
   3) Welchen Unterschied gibt es zwischen den Ausdrücken `3*4+5` und `5+4*3`?
   4) Welchen Unterschied gibt es zwischen den Ausdrücken `(3*4)+5` und `3*(4+5)`?

b) Schreiben Sie folgende Formeln in `Modula-3`-Ausdrücke um:

1) $z = 1 + a \cdot \frac{b}{c}$
2) $y = \frac{a+b}{c} \cdot (d - e) \cdot f$
3) $w = (a - b)^2$
4) $v = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}$

6 Übung: Numerische und ordinale Typen

Aufgabe 23 – Typenwahl

Mit welchen Typen lassen sich die folgenden Daten am besten beschreiben? Schreiben Sie entsprechende Typdefinitionen in `Modula-3`.

- Postleitzahl
- Bild einer Rommékarte (also Sieben, Dame, König, usw.)
- Farbe einer Rommékarte
- Nummer eines Stiches beim Skat
- Ampelfarbe
- Ampelzustand (alle möglichen Zustände, nicht nur die erlaubten)
- Etage im MZH
- Masse
- Teilnehmeranzahl für eine Vorlesung
- Anwesenheitsliste für eine kleine und feste Personengruppe
- Kontostand
- Geldscheinart
- Schachfigur
- Monat
- Jahreszeit, genauer: die Monate die zu einer Jahreszeit gehören

Aufgabe 24 – Zahlenein- und Ausgabe

Schreiben Sie ein Programm, das Sie nach drei Zahlen fragt und dann die Summe der drei Zahlen ausgibt. Nachdem die Summe ausgegeben wurde, soll nach einer weiteren Zahl gefragt werden, mit der die Summe dann multipliziert wird. Dieses Ergebnis soll ebenfalls ausgegeben werden.

Aufgabe 25 – Logische Operatoren

a) Wie würden Sie folgende Bedingung beschreiben: `(x AND NOT y) OR (NOT x AND y)`?

b) Worin liegt der Unterschied zwischen `a AND b` und `NOT (NOT a OR NOT b)`?

c) Wie würden Sie den Ausdruck `NOT a` ohne den **NOT**-Operator formulieren?

d) Welche Klammern sind hier überflüssig: `(a AND b) OR ((NOT c) AND (d OR e))`?

Aufgabe 26 – Noch mehr logische Operatoren

Mit der Deklaration

```
CONST a=29; b=3; c=5;
```

ergeben sich in den folgenden Ausdrücken welche logischen Werte?

a) `10 < 3 AND 17 > 4`

b) `17 + 4 = 21 OR c`

c) `a # 0`

d) `a = 0 OR b > 5`

e) `a = 0 AND b > 5`

f) `a > 0 OR (a < 10 AND b = 1)`

g) `(a > 0 OR a < 10) AND b = 1`

h) `a > 0 OR a < 10 AND b = 1`

7 Übung: Auswahlstrukturen und Sichtbarkeit

Aufgabe 27 – Bedingte Anweisungen

a) Ein **IF**-Anweisungsblock wird immer nur dann ausgeführt, wenn die Bedingung wahr ist. Ist diese Aussage wahr oder falsch?

b) Es ist unmöglich, eine **IF**-Anweisung so zu programmieren, dass der Anweisungsblock ausgeführt wird, wenn die Bedingung falsch ist. Ist diese Aussage wahr oder falsch?

c) Kann man auf **ELSIF** verzichten? Wie kann man `IF A THEN B ELSIF C THEN D END;` ohne **ELSIF** schreiben?

d) Kann man `IF A THEN B ELSE C END;` so umwandeln, dass auf das **ELSE** verzichtet werden kann?

e) Wie kann man `IF A AND B THEN C END;` so umwandeln, dass **AND** nicht benötigt wird?

f) Wie kann man `IF A OR B THEN C END;` so umwandeln, dass **OR** nicht benötigt wird?

g) Nutzen Sie die gefundenen Umformungen (gegebenenfalls auch in umgekehrter Richtung) um folgendes Programmfragment in einen einzigen **IF**-Block ohne Vergleiche zu vereinfachen.

```
IF A = FALSE THEN
  BEGIN
    IF B = TRUE THEN
      C;
    ELSE
      IF D # FALSE THEN
        C;
      END;
    END;
  END;
END;
```

Aufgabe 28 – Geschachtelte bedingte Anweisungen

Es seien `a,b,c,d` und `stat` als Ganzzahl-Variablen vereinbart. Ein hoffentlich abschreckendes Beispiel für schlechte Formatierung stellt folgende geschachtelte **`IF`**-Anweisung dar:

```
IF a < b THEN IF c < d THEN             stat := 1;
  ELSE IF a < c THEN IF b < d THEN      stat := 2;
                                   ELSE stat := 3;
  END ELSE IF a < d THEN IF b < c THEN stat := 4;
                                   ELSE stat := 5;
                              END ELSE stat := 6;
                  END END END      ELSE stat := 7;
  END
```

a) Bestimmen Sie die Werte, die der Variablen `stat` in den folgenden vier Fällen zugewiesen werden:

Fall	a	b	c	d	stat
1)	0	1	1	0	
2)	1	0	0	1	
3)	0	1	0	1	
4)	1	0	1	0	

b) Welche der auftretenden Vergleiche sind in der obigen **`IF`**-Anweisung überflüssig (d.h. ihr Wert ist bereits durch die Auswertung vorangegangener Vergleiche eindeutig festgelegt oder sie werden überhaupt nie ausgeführt)?

1) `a < b` 3) `c < d` 5) `a < c`

2) `b < c` 4) `a < d`

c) Bestimmen Sie die minimale Anzahl von Vergleichen, die in einer verbesserten geschachtelten **`IF`**-Anweisung auftreten müssen, die denselben Effekt erzielt wie die obige Anweisung.

Aufgabe 29 – Fallunterscheidung

Definieren Sie einen Aufzählungstypen für Monate. Schreiben Sie eine **`CASE`**-Anweisung, die für jeden Monat die Länge in Tagen ausgibt.

Aufgabe 30 – Bezugsrahmen von Variablen

a) 1) Ist es möglich, an den Inhalt einer lokalen Variablen einer anderen Funktion heranzukommen?

2) Können eine lokale und eine globale Variable den gleichen Namen haben?

3) Unter welchen Umständen können zwei lokale Variablen den gleichen Namen haben?

4) Können eine Konstante und eine Variable den gleichen Namen haben?

b) Gegeben sei folgendes Programmfragment:

```
VAR
  a, b: INTEGER;
...                                  (*1*)
PROCEDURE X() =
  VAR x1: INTEGER;
  BEGIN
    ...                              (*2*)
  END X;

PROCEDURE Y() =
  VAR y1, a, x1: INTEGER
  BEGIN
    ...                              (*3*)
    VAR a, z: INTEGER;
    BEGIN
      ...                            (*4*)
    END;
  END;
...                                  (*5*)
```

1) Welche Objekte (Variablen, Funktionen) sind an den Punkten `(*1*)`, `(*2*)`, `(*3*)`, `(*4*)`, `(*5*)` ansprechbar?

2) Welche Variablen sind in der Funktion `Y` global und welche lokal?

8 Übung: Unterprogramme

Aufgabe 31 – Logische Operatoren im Eigenbau

Gegeben sei die Funktion

```
PROCEDURE And (x, y: BOOLEAN; ): BOOLEAN =
  BEGIN
    RETURN x AND y;
  END And;
```

Was ist der Unterschied zwischen

- `A AND B` und
- `And(A,B)` ?

Aufgabe 32 – **`RETURN`** mit Logik

- Was ist der Unterschied zwischen

```
IF A THEN
  RETURN TRUE;
ELSE
  RETURN FALSE;
END;
```

und

```
IF A THEN
  RETURN TRUE;
END;
RETURN FALSE;
```

?

- Wie kann man die Anweisungsfolgen vereinfachen?

Aufgabe 33 – Zahlentheorie

- Was bewirkt folgende Funktion?

```
PROCEDURE Test (x: INTEGER; ): BOOLEAN =
  VAR
    ergebnis: BOOLEAN;
  BEGIN
    ergebnis := x MOD 2 = 0;
    IF ergebnis = TRUE THEN
      RETURN TRUE;
    END;
    RETURN FALSE;
  END Test;
```

  Geben Sie ihr einen treffenden Namen.

- Vereinfachen Sie die Funktion. Man benötigt keine Variable und kommt zwischen **BEGIN** und **END** mit einer **RETURN**-Anweisung und einem Ausdruck aus!

- Verallgemeinern Sie die Funktion und rufen Sie nun von der speziellen Funktion die allgemeine auf.

Aufgabe 34 – Schaltjahre

Schreiben Sie eine Funktion, die zu einer Jahreszahl die Anzahl der Schalttage ermittelt. Die Bedingung „`year` ist ein Schaltjahr“ lässt sich in einem einzigen Ausdruck formulieren. Die Funktion sollte folgende Signatur haben:

```
PROCEDURE NumberOfIntercalaryDays
   (year : CARDINAL; ) : [0..1];
```

Schreiben Sie mit dieser Funktion einen Ausdruck, der die Anzahl der Tage des entsprechenden Jahres berechnet.

Zum Hintergrund: Zunächst sind alle durch 4 teilbaren Jahreszahlen Schaltjahre. Ausnahmen sind lediglich die Säkularjahre „XX00“ die nicht durch 400 teilbar sind: So war weder 1800 noch 1900 ein Schaltjahr, 2000 hingegen war wieder ein Schaltjahr.

Warum ist das so? Ein Sonnenjahr beträgt 365,2422 Tage. Dieser Überschuss von 0,2422 Tagen wird durch den zusätzlichen Tag im Schaltjahr nur ungenau ausgeglichen. Durch ihn entsteht wiederum ein kalendarisches Defizit von 0,0312 Tagen in vier Jahren, also von 3,12 Tagen in 400 Jahren. Diese 3,12 Tage gleicht wiederum die Regelung, dass nur jedes vierte Säkularjahr ein Schaltjahr ist, weitgehend aus. Die Restdifferenz in 400 Jahren beträgt dann nur noch 0,12 Tage.

9 Übung: Schleifen

Aufgabe 35 – **FOR**-Schleifen

Erklären Sie, was folgende Konstrukte bewirken:

a)
```
FOR x:=1 TO 100 DO S END;
```

b)
```
FOR x:=2 TO 1 DO S END;
```

c)
```
FOR x:=2 TO 1 BY -1 DO S END;
```

d)
```
FOR x:=1 TO 10 BY 4 DO S END;
```

Aufgabe 36 – **REPEAT**, **WHILE**, **LOOP**

a) Welche Auswirkungen hat **WHILE TRUE DO END;** auf den Programmfluss? Mit welcher Schleife erreicht man das gleiche?

b) Ist es möglich, ein **REPEAT** ... **UNTIL** nur mit **WHILE** zu simulieren oder umgekehrt? Wenn ja, wie?

Aufgabe 37 – Größter gemeinsamer Teiler

Schreiben Sie eine Funktion namens `GGT`, der Sie zwei ganzzahlige Werte übergeben, und die den größten gemeinsamen Teiler der beiden Zahlen zurückgibt (also die größte Zahl, durch die beide Zahlen ohne Rest teilbar sind).

Programmieren Sie zum einen den in der Vorlesung angegebenen EUKLIDischen Algorithmus und verwenden Sie zum anderen folgende Idee: Es sei a die kleinere der beiden Zahlen. Dann gilt für den größten gemeinsamen Teiler t: $1 \leq t \leq a$. Daher teste man zunächst, ob a der größte gemeinsame Teiler ist. Wenn nicht, dann verringere a um 1 und teste erneut, usw.

Welche Methode ist günstiger?

Aufgabe 38 – Summe von Quadratzahlen

a) Schreiben Sie eine Funktion namens `Sum` zur Addition der ersten 100 Quadratzahlen. Die Funktion soll dabei Variablen vom Typ **CARDINAL** benutzen.

b) Modifizieren Sie die Funktion so, dass die Anzahl der aufzusummierenden Quadratzahlen an die Funktion übergeben werden kann.

10 Übung: Felder und Texte

Aufgabe 39 – Monatslängen

In Aufgabe 29 haben wir mit der **CASE**-Struktur jedem Monat seine Länge in Tagen zugeordnet. Mit Feldern geht dies wesentlich eleganter. Legen Sie eine Konstante vom Typ **ARRAY** `Month` **OF** `[28..31]` an, in der jedem Monat seine Länge zugeordnet wird.

Schreiben Sie eine Funktion, die diese Konstante lokal definiert, und welche zu gegebenem Jahr und Monat die Anzahl der Tage des Monats berechnet.

Aufgabe 40 – Aufzählungselemente in Texte konvertieren

a) Definieren Sie einen Aufzählungstyp für alle Wochentage.

b) Definieren Sie ein konstantes Feld, welches jedem Wochentag (Element des obigen Aufzählungstyps) den Namen des Wochentages zuordnet.

Aufgabe 41 – Maximumsbestimmung

Schreiben Sie eine Funktion `Maximum`, der man beliebig viele ganze Zahlen übergeben kann, und die dann den größten der n Werte zurückgibt.

Aufgabe 42 – Mengen

a) Definieren Sie einen Aufzählungstyp für die drei Farben einer Ampel.

b) Definieren Sie ein konstantes Feld, welches der Reihenfolge nach die Ampelphasen Rot, Rot-Gelb, Grün, Gelb enthält.

c) Schreiben Sie eine Funktion, die für eine willkürliche Kombination dieser Farben die Farbzusammenstellung als Text zurückgibt.

d) Wie kann man für zwei Farbkombinationen effizient testen, ob mindestens eine Farbe in beiden Kombinationen vorkommt?

Aufgabe 43 – Manipulation von Zeichenketten

a) Schreiben (und testen) Sie eine Funktion `Upper`, die alle Kleinbuchstaben einer Zeichenkette in Großbuchstaben verwandelt, Großbuchstaben und andere Zeichen aber unverändert lässt.
Verwandeln Sie zunächst den Text mit `Text.SetChars` in ein Feld von Zeichen. Benutzen Sie zur Konvertierung die Menge aller Kleinbuchstaben `ASCII.Lowers` und das Feld `ASCII.Upper`, das allen Zeichen den entsprechenden Großbuchstaben, oder falls nicht existent, das Zeichen selbst zuordnet. Konvertieren Sie das Zeichenfeld zum Schluss mit `Text.FromChars` wieder zurück in einen Text.

b) Schreiben und testen Sie eine Funktion `Reverse`, die eine Zeichenkette umdreht (d.h. das ursprünglich erste Zeichen steht dann an letzter Stelle, das zweite an vorletzter Stelle usw.) und zurückgibt.

11 Übung: Datenverbünde, Zeiger, Funktionsvariablen

Aufgabe 44 – Zeiger und Felder

a) Worin liegt der Unterschied zwischen

   1) `ARRAY [0..0] OF INTEGER` und `INTEGER`,
   2) `REF ARRAY OF INTEGER` und `ARRAY OF REF INTEGER`?

b) Es gibt verschiedene Modi bei der Übergabe eines Feldes an eine Funktion. Angenommen, Sie wollen innerhalb der Funktion Änderungen an dem Feld vornehmen, die nicht global wirksam werden sollen, d.h. nach Verlassen der Funktion soll das Feld in der aufrufenden Funktion unverändert sein. Wie lässt sich das realisieren? Angenommen Sie wollen keine Änderungen vornehmen, welcher Modus ist dann besser und warum?

c) Angenommen, Sie haben ein dreidimensionales Feld `ddd` der Größe $n_1 \times n_2 \times n_3$ und ein eindimensionales Feld `d` mit gleich vielen Elementen. Wie können Sie `ddd` mit `d` simulieren, also wie können Sie die Werte aus `ddd` in `d` anordnen, und was ist dann das Gegenstück zum Feldzugriff `ddd[i,j,k]`?

d) Die doppelte Dereferenzierung `x^^` ist Ihnen geläufig. Ist die wiederholte Adressbestimmung `ADDRESS(ADDRESS(x))` ebenfalls sinnvoll?

e) Es seien `a` und `b` kompatible Zeiger. Was ist der Unterschied zwischen den Zuweisungen `a := b;` und `a^ := b^`?

Aufgabe 45 – Datenverbünde und Funktionen

a) Wie kann eine Funktion mehrere Werte zurückgeben?

b) Definieren Sie einen Datenverbund für komplexe Zahlen. Schreiben Sie eine Funktion, die zwei komplexe Zahlen multipliziert und eine komplexe Zahl als Ergebnis liefert. Für die Multiplikation komplexer Zahlen gilt

$$(a + i \cdot b) \cdot (c + i \cdot d) = (a \cdot c - b \cdot d) + i \cdot (a \cdot d + b \cdot c)$$

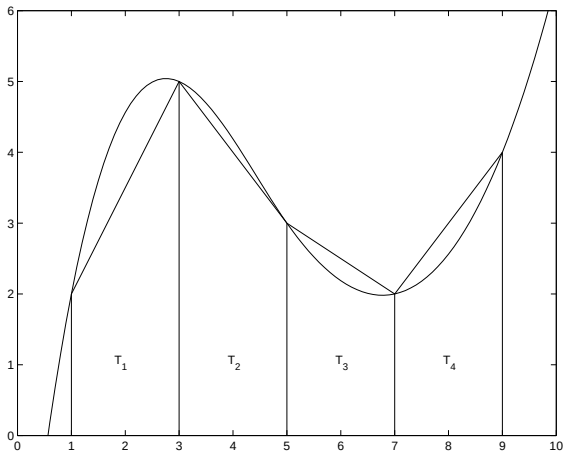


Abbildung 1: Trapezregel

Aufgabe 46 – Numerische Integration

Für die numerische Berechnung von Integralen geht man auf die Grundidee der Ausschöpfung des Flächeninhalts zurück. Statt Rechtecken sollen hier Trapeze verwendet werden:

Definiert man

$$t_k = a + k \cdot \frac{b-a}{n} \quad \text{für } k = 0, \ldots, n \,,$$

dann hat das k-te Trapez mit den Eckpunkten $t_{k-1}, t_k, f(t_{k-1})$ und $f(t_k)$ den Flächeninhalt

$$T_k = (t_k - t_{k-1}) \cdot (f(t_{k-1}) + f(t_k))/2$$

und als Approximation an den Gesamtinhalt ergibt sich

$$T^{(n)} \;=\; \sum_{k=1}^{n} T_k = \sum_{k=1}^{n} (t_k - t_{k-1}) \cdot \frac{f(t_{k-1}) + f(t_k)}{2} = \frac{b-a}{n} \cdot \left(\frac{1}{2} f(a) + \sum_{k=1}^{n-1} f(t_k) + \frac{1}{2} f(b) \right) \,.$$

a) Programmieren Sie eine `Modula-3`-Funktion `Trapez`, der Sie die Intervallgrenzen a, b, die Anzahl n der Trapeze und die zu integrierende Funktion f übergeben, und die dann $T^{(n)}$ zurückgibt.

b) Testen Sie Ihre Funktion anhand der Integrale

$$\int_{-2\pi}^{2\pi} \frac{\sin(t)}{t} \, \mathrm{d}t \,, \qquad \int_{0}^{1} x^{20} e^{x-1} \, \mathrm{d}x \,.$$

Was sind vernünftige Größenordnungen für die Anzahlen der Trapeze?

12 Übung: Umgang mit Dateien und Fehlerbehandlung

Aufgabe 47 – Umgang mit Dateien

Schreiben Sie ein Programm, das Sie nach dem Namen einer Quelldatei und dem einer Zieldatei (jeweils Textdateien) fragt. Der Text, der in der Quelldatei steht, soll zeilenweise umgedreht werden und in der Zieldatei abgespeichert werden.

BEISPIEL: Der Inhalt einer Datei

NEBEL vorwaerts wird
zu rueckwaerts LEBEN

wird zu

driw streawrov LEBEN
NEBEL streawkceur uz

Aufgabe 48 – Zahleneingabe

Schreiben Sie eine Funktion, die vom Benutzer eine Gleitkommazahl einliest. Gibt der Benutzer keine Zahl ein, soll er auf den Fehler hingewiesen und um eine neue Zahl gebeten werden. Der Funktion soll man den Eingabeaufforderungstext übergeben können und die maximale Anzahl der Fragewiederholung. Sehen Sie ein Schlüsselwort vor, mit dem der Benutzer die Fragerei abbrechen kann. Im Falle eines Abbruchs durch den Benutzer oder wegen der Begrenzung der Rückfragen, kann die Funktion keine Zahl zurückgeben. Deswegen soll sie in diesem Falle eine eigene Ausnahme auslösen.

13 Übung: Module und Objekte

14 Übung: Zusatzaufgaben

Aufgabe 49 – Programmfehler

Was ist in folgenden Programmen falsch?

Programm 1:

```
MODULE Main;

FROM IO IMPORT Put;

PROCEDURE Test1() =
  BEGIN
    Put("Dies ist der erste Test\n");
  END Test1

PROCEDURE Test2(b: BOOLEAN; ) =
  BEGIN
    Put("Dies ist der zweite Test\n");
  END Test2

BEGIN
  Test1();
  Test2(1);
END Main;
```

Programm 2:

```
MODULE Main;

IMPORT Put From IO;

BEGIN
  Ausgabe;
END Main.

PROCEDURE Ausgabe();
  BEGIN
    Put (Dies ist eine Testausgabe);
  END;
```

Programm 3:

```
MODULE Main;

IMPORT IO, Stdio, Fmt, Lex;

VAR
  a: CARDINAL;

BEGIN
  IO.Put ("Bitte geben Sie eine negative Zahl ein:");
  a := Lex.Int (Stdio.stdin);
  IO.Put (Fmt.F ("Die Zahl lautet %s\n", Fmt.Int(a)));
END Main.
```

Falls Sie das Programm starten, wird der Fehler nicht während der Übersetzung, sondern während der Ausführung auftreten.

Aufgabe 50 – Fehlermeldungen

Es ist wichtig zu wissen, wie der Übersetzer auf verschiedene Fehler reagiert. Schreiben Sie je ein Programm welches folgende Fehler enthält und studieren Sie die Fehlermeldungen des Übersetzers.

a) vergessenes Semikolon

b) vergessene Anführungszeichen

c) vergessenes Ende eines Kommentars

d) Deklaration einer Variable mit dem Namen eines importierten Moduls

e) eine Funktion (mit Rückgabewert) ohne **RETURN** `wert`

f) **RETURN** `wert` in einer Prozedur (ohne Rückgabewert)

g) Aufruf einer Prozedur innerhalb eines mathematischen Ausdruckes

h) Aufruf einer Funktion wie eine Anweisung

i) ein unbehandelter Fehler (z.B. mit `Lex.Int`)

Aufgabe 51 – Vorzeichenfunktion

Schreiben Sie eine Funktion namens `Sign`, der man einen Integer-Wert übergibt, und die folgendes zurückliefert: -1, wenn der Wert negativ ist, 0, wenn der Wert Null ist und 1, wenn der Wert positiv ist. Zeigen Sie dem Anwender der Funktion `Sign`, dass sie wirklich nur die Werte -1, 0 und 1 zurückgeben kann.

Aufgabe 52 – Potenzfunktion

a) Schreiben Sie eine Funktion namens `Power`, die Sie z.B. mit `Power(x,n)` aufrufen können und die dann x^n berechnet. Die Basis soll dabei vom Typ **`LONGREAL`** und der Exponent vom Typ **`INTEGER`** sein.

b) Die naheliegende Implementation der `Power`-Funktion benötigt $|n| - 1$ Multiplikationen und gegebenenfalls eine Division. Der Rechenaufwand lässt sich drastisch reduzieren auf maximal $2 \log_2 |n|$ Multiplikationen und eine Division. Wie?

Aufgabe 53 – Fehlerwahrscheinlichkeiten

a) Angenommen ein Programm enthält 30 Fehler von denen jeder mit der Wahrscheinlichkeit 10^{-6} auftritt, das heißt, dass jeder Fehler pro eine Million Programmaufrufe durchschnittlich einmal zuschlägt. Das Programm wird von 1000 Benutzern jeweils 100mal gestartet. Wie hoch ist die Wahrscheinlichkeit, dass bei einem der Benutzer ein Fehler zu Tage tritt?

b) Schreiben Sie eine Funktion, die die Wahrscheinlichkeit für den fehlerfreien Ablauf eines Programmes bei k Aufrufen bei l Benutzern berechnet, wenn das Programm n Fehler mit Auftrittswahrscheinlichkeit $\frac{1}{m}$ enthält.

Aufgabe 54 – Primzahltest

a) Schreiben Sie eine Funktion namens `IsPrime`, der Sie eine natürliche Zahl n übergeben und die **`TRUE`** zurückgibt, wenn n eine Primzahl ist, und **`FALSE`** zurückgibt, wenn n Zahl keine Primzahl ist. (Achtung: 1 ist keine Primzahl!)

b) Beachten Sie, dass man für diesen Test verglichen mit der Größe von n recht wenig Divisionen benötigt. Überlegen Sie sich, welche Divisionen man sparen kann und optimieren Sie ihr Programm dahingehend.

Aufgabe 55 – Fakultätsfunktion

Die Fakultät einer Zahl N, geschrieben $N!$, ist definiert als:

$$N! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot N \quad \text{für} \quad N \geq 1$$
$$N! = 1 \quad \text{für} \quad N = 0$$

Welches ist die größte ganze Zahl, für die $N!$ auf dem benutzten Rechner als **CARDINAL**-Größe exakt darstellbar ist?

Man löse dieses Problem nicht durch heftiges Hinsehen, sondern mithilfe eines Programms, welches diese Zahl ermittelt. Man verwende bei der Berechnung nur Variablen vom Typ **CARDINAL**.

Aufgabe 56 – Noch mehr Schleifen

```
1  MODULE Main;
2  IMPORT IO, Fmt;
3  VAR
4    sum, i: CARDINAL;
5  BEGIN
6    sum := 0;
7    i := 1;
8    WHILE i <= 100 dO
9      sum := sum + 1;
10     i := i + 1;
11   END;
12   IO.Put(F("i=%s, summe=%s\n", Fmt.Int(i), Fmt.Int(sum)));
13 END Main.
```

In diesem Programm sollen die Zeilen 6 bis 11 ersetzt werden. Welche Anweisungsfolgen sind geeignet, damit das Programm genau das gleiche leistet?

a)
```
sum := 0; i := 1;
REPEAT
  sum := sum + i;
  i := i + 1;
UNTIL i = 101;
```

b)
```
sum := 0;
FOR i := 1 TO 100 DO
  sum := sum + 1;
END;
```

c)

```
sum := 0;
i := 1;     sum := sum + i;
i := 2;     sum := sum + i;
  ...
i := 100;   sum := sum + i;
```

d)

```
sum := 0;
sum := sum + 1;
sum := sum + 2;
  ...
sum := sum + 100;
```

e)

```
sum := 0; i := 0;
REPEAT
  i := i + 1;
  sum := sum + 1;
UNTIL i = 101;
```

Aufgabe 57 – Kleinstes gemeinsames Vielfaches

Schreiben Sie eine Funktion namens `KGV`, der Sie zwei ganzzahlige Werte übergeben, und die das kleinste gemeinsame Vielfache der beiden Zahlen zurückgibt (also die kleinste Zahl, die durch beide Zahlen ohne Rest geteilt werden kann).

Aufgabe 58 – Logarithmus-Funktion

Schreiben Sie eine Funktion namens `LogX`, der Sie zwei Zahlen a und b übergeben, und die den Logarithmus von a zur Basis b berechnet.

Aufgabe 59 – Wertetabelle

Gegeben sei die abgebrochene Taylor-Reihe der Exponentialfunktion `exp`:

$$f(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4$$

Man lasse eine Wertetabelle dieser Funktion im Intervall $[0, 1]$ ausdrucken. Was ist die sparsamste und sicherste Methode, $f(x)$ zu programmieren?

Aufgabe 60 – Harmonische Reihe

Man schreibe zwei Funktionen, die die n. harmonische Zahl H_n mit

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

berechnen, wobei die eine Funktion in obiger und die andere umgekehrter Reihenfolge addieren soll. Man vergleiche die Ergebnisse. Welches ist besser und warum?

Aufgabe 61 – Quadratische Gleichungen

Man schreibe ein Programm zur Lösung der allgemeinen quadratischen Gleichung

$$a \cdot x^2 + b \cdot x + c = 0.$$

Nach der Transformation (falls $a \neq 0$)

$$b \to \frac{b}{a} = p\,,\; c \to \frac{c}{a} = q$$

lautet die Lösung

$$x_{1/2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \quad (\text{falls } \frac{p^2}{4} \geq q)$$

bzw.

$$x_{1/2} = -\frac{p}{2} \pm i\sqrt{q - \frac{p^2}{4}} \quad (\text{falls } \frac{p^2}{4} < q)$$

(mit $i^2 = -1$). Für $a = 0, b \neq 0$ lautet die Lösung $x = -\frac{c}{b}$. Falls $b = 0$ und $a = 0$, gibt es nur eine Lösung, wenn auch $c = 0$ gilt.

Man programmiere sorgfältig sämtliche Fallunterscheidungen.

Aufgabe 62 – Maschinengenauigkeit

Als Maschinengenauigkeit eines Rechners bezeichnet man die kleinste Maschinenzahl e für die auf dem Rechner gilt:

$$1 + e > 1$$

Diese kleinste Zahl e wird häufig auch mit EPS bezeichnet. Also:

$$\text{EPS} = \min\{e : 1 + e > 1\}$$

Man schreibe ein Programm, das EPS berechnet.

Aufgabe 63 – Babylonisches Wurzelziehen

Die alten Babylonier besaßen zwar noch keine elektronischen Rechenanlagen, dafür aber ein Verfahren, die Wurzel aus einer positiven Zahl zu ziehen. Der Algorithmus 'Babylonisches Wurzelziehen' sieht so aus:

Gegeben sei eine nichtnegative reelle Zahl C und eine beliebige positive reelle Zahl x_0 (z.B.: $x_0 = 1$). Dann definiere die Folge

$$\begin{aligned} x_1 &= \frac{1}{2}\left(x_0 + \frac{C}{x_0}\right) \\ x_2 &= \frac{1}{2}\left(x_1 + \frac{C}{x_1}\right) \\ &\vdots \\ x_{n+1} &= \frac{1}{2}\left(x_n + \frac{C}{x_n}\right) \end{aligned}$$

Schon nach wenigen Gliedern liefert diese Folge eine gute Näherung an $\sqrt{C}$; weitere Folgenglieder unterscheiden sich nur sehr wenig untereinander. Zur Vermeidung einer Endlosschleife sollte ein Abbruchkriterium in Form der folgenden Abfrage formuliert werden: Falls für eine vorgegebene Konstante K

$$|x_{n+1} - x_n| \leq K \cdot |x_{n+1}|$$

gilt, dann brich die Iteration ab. Als Konstante bieten sich z.B. `5.0E-6`, `1.0E-6` oder EPS (siehe Aufgabe 62) an. Warum ist 0 nicht geeignet?

Man schreibe eine Funktion `Wurzel` und teste sie gegen die Standard-Funktion `Math.sqrt`.

Aufgabe 64 – Berechnung von π

Für ganzzahliges n seien a_n, b_n die Seitenlängen eines dem Einheitskreis ein- bzw. umbeschriebenen regelmäßigen $6 \cdot 2^n$-Ecks. Dann gilt:

$$6 \cdot 2^{n-1} a_n \leq \pi \leq 6 \cdot 2^{n-1} b_n \,.$$

Geometrische Überlegungen führen auf folgende Ausdrücke:

$$\begin{aligned} a_0 &= 1, \quad b_0 = \frac{2}{\sqrt{3}} \\ a_{n+1} &= \sqrt{2(1 - \sqrt{1 - (\frac{a_n}{2})^2})} \\ a_{n+1} &= \sqrt{2\frac{1 - (1 - (\frac{a_n}{2})^2)}{1 + \sqrt{1 - (\frac{a_n}{2})^2}}} \\ a_{n+1} &= \sqrt{2\frac{(\frac{a_n}{2})^2}{1 + \sqrt{1 - (\frac{a_n}{2})^2}}} \\ a_{n+1} &= \frac{a_n}{\sqrt{2(1 + \sqrt{1 - (\frac{a_n}{2})^2})}} \end{aligned}$$

$$
\begin{aligned}
b_{n+1} &= 4\frac{\sqrt{(\frac{b_n}{2})^2+1}-1}{b_n} \\
b_{n+1} &= 4\frac{((\frac{b_n}{2})^2+1)-1}{b_n(\sqrt{(\frac{b_n}{2})^2+1}+1)} \\
b_{n+1} &= 4\frac{(\frac{b_n}{2})^2}{b_n(\sqrt{(\frac{b_n}{2})^2+1}+1)} \\
b_{n+1} &= \frac{b_n}{\sqrt{(\frac{b_n}{2})^2+1}+1}
\end{aligned}
$$

a) Rechnen Sie nach, dass sich die verschiedenen Ausdrücke für a_{n+1} bzw. b_{n+1} ineinander überführen lassen.

b) Berechnen Sie für $n = 1, \ldots, 100$ nach jeder dieser Formeln untere und obere Schranken für π.

c) Vergleichen und erklären Sie die Ergebnisse.

Aufgabe 65 – Rekursion

Schreiben Sie eine rekursive Funktion mit der Signatur

```
PROCEDURE PutDigit (wert: INTEGER;);
```

die die Zahl `wert` zeichenweise in richtiger Reihenfolge ausgibt. Vergessen Sie nicht die Ausgabe des Vorzeichens!

Aufgabe 66 – Ackermann-Funktion

Für natürliche Zahlen m, n (einschließlich 0) definiert

```
PROCEDURE Acker(m, n: CARDINAL; ): CARDINAL =
  BEGIN
    IF m = 0 THEN
      RETURN n+1;
    ELSIF n = 0 THEN
      RETURN Acker(m-1, 0);
    ELSE
      RETURN Acker(m-1, Acker(m, n-1));
    END;
  END Acker;
```

die sogenannte Ackermann-Funktion.
Welchen Wert hat `Acker(2,2)`?

Aufgabe 67 – Fibonacci-Zahlen

a) Schreiben Sie eine Funktion namens `Fibonacci`, die die Fibonaccizahlen berechnet (als **`CARDINAL`** und ausdruckt:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_2 &= f_1 + f_0 = 1 + 0 = 1 \\ f_3 &= f_2 + f_1 = 1 + 1 = 2 \\ f_4 &= f_3 + f_2 = 2 + 1 = 3 \\ &\vdots \\ f_{n+1} &= f_n + f_{n-1} \end{aligned}$$

b) Modifizieren Sie das Programm so, dass die zu vorgegebenem n berechneten Fibonaccizahlen in einem eindimensionalen Feld abgespeichert werden. Dieses soll dann an die aufrufende Funktion zurückgegeben werden.

c) Berechnen Sie die Fibonaccizahlen mit einer rekursiven Funktion und vergleichen Sie die Ausführungszeiten.

Aufgabe 68 – Die Türme von Hanoi

Kennen Sie das Spiel „Die Türme von Hanoi“? Es besteht aus drei Stangen A, B und C sowie aus n Scheiben unterschiedlichen Durchmessers, die der Größe nach gestapelt auf der Stange A einen Turm bilden. Aufgabe ist es, diesen Turm von A nach C zu transportieren, wobei

- immer nur eine Scheibe auf einmal bewegt werden darf,
- nur die oberste Scheibe von einem Turm genommen werden kann,
- nie eine größere auf einer kleineren Scheibe liegen darf.

Programmieren Sie eine rekursive Funktion mit Eingabeparameter n, die diese Aufgabe löst. Dabei soll jeweils ausgegeben werden, von welcher Stange zu welcher Stange eine Scheibe bewegt wurde.
Wieviele Funktionsaufrufe (in Abhängigkeit von n) sind nötig?

Hinweis: Die Lösung ist einfach für $n = 1$: Lege diese Scheibe von A nach C. Ist mehr als eine Scheibe vorhanden, muss man die Stange B zuhilfe nehmen, z.B. für $n = 2$: Lege die kleinere Scheibe von A nach B, dann die untere von A nach C und schließlich die kleinere von B nach C.

Aufgabe 69 – Pascalsche Dreiecke

Mithilfe des Pascalschen Dreiecks lassen sich bekanntlich die Koeffizienten $\binom{n}{0}, \ldots, \binom{n}{n}$ des Binomialausdrucks

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

leicht berechnen:

$$\binom{0}{0} = 1, \quad \binom{n}{0} = \binom{n}{n} = 1, \quad \binom{n}{i} = \binom{n-1}{i-1} + \binom{n-1}{i} \text{ für } i \in \{1, \ldots, n-1\}.$$

Schreiben Sie ein `Modula-3`-Programm, welches die ersten m Zeilen des PASCALschen Dreiecks für ein vorzugebendes m in einem dreieckigen Feld vom Typ **`ARRAY OF REF ARRAY OF INTEGER`** zurückgibt und wie im Beispiel unten ausgibt.

BEISPIEL: Für $m = 4$ ergibt sich folgende Ausgabe:

```
n =  0:   1
n =  1:   1   1
n =  2:   1   2   1
n =  3:   1   3   3   1
n =  4:   1   4   6   4   1
```

15 Übung: Aufgaben auf Halde

Aufgabe 70 – Fallunterscheidung

Schreiben Sie ein Programm, welches vom Benutzer zwei **`LONGREAL`**-Zahlen und ein einzelnes Operationszeichen einliest. Für die Operationszeichen '+', '-', '*', '/' sollen die entsprechenden Operationen ausgeführt werden, andernfalls soll eine Fehlermeldung ausgegeben werden.

Aufgabe 71 – Funktionen mit Parameterübergabe

a) Es wird gesagt, dass Funktionsaufrufe überall dort stehen können, wo auch Variablen stehen können. Wo aber liegt der Unterschied zwischen dem Wert einer Variablen und dem Wert einer Funktion?

b) Worin liegt der Unterschied zwischen einem Funktionsparameter und einer lokalen Variablen dieser Funktion?

c) Hat das Ändern des Wertes eines Funktionsparameters innerhalb einer Funktion Auswirkungen auf die Variable im Funktionsaufruf?