

Einführung

Sonderübung

Nadine Schärmann & Thomas Klütz

PR10

2. November 2020

1. Organisatorisches
 - Vorwissen und Ablauf
 - Verweise
 - Shell/ Bash/ Command Prompt
 - Editor
2. Unser erstes FORTRAN-Programm
 - Fortran95 - eine Programmiersprache lernen
 - Hello World
3. Einfache Ein- und Ausgabe auf der Konsole
 - Übungsaufgabe
 - Der GNU Fortran Compiler
4. Übungsaufgaben

Über diesen Kurs

Vorwissen

- Ihr habt bereits einen Computer benutzt.
- Ihr habt hoffentlich bereits einen Compiler und einen Editor installiert.
- Eventuell habt ihr schon eine (andere) Programmiersprache kennengelernt.

Ablauf

- Insgesamt haben wir ca. 12 Stunden.
- Jede Stunde behandelt ein Thema und liefert Übungen, um das neu gewonnene Wissen anzuwenden.
- Fragen zu den aktuellen Hausaufgaben können besprochen werden
- Die Sonderübung dient als Ergänzung zum normalen Übungsbetrieb.

Einige Quellen

- Ihr könnt eure Tutoren fragen
- Opal-Kurs zur Vorlesung:
<https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/26811596802/?17>
- Fortran 95 - Nachschlagewerk vom RRZN Hannover
- Material-Repository:
<https://gitlab.mn.tu-dresden.de/wwalter/PROG-material-public>

Liste wichtiger Befehle

- Worte in eckigen Klammern (<>) sind Platzhalter für Parameter
- Syntax ist <befehl> <parameter>
- Eingabe von Parametern entweder wie file names oder mittels dash (–) vorneweg

Linux	Windows	
man <befehl>	help <befehl>	öffnet das Manual (Handbuch) zum entsprechenden Befehl (schließe mit q)
ls	dir	listet den Inhalt des Directories auf
cd <dir>	cd <dir>	geht in das Directory <dir>
mkdir <dir>	md <dir>	erstellt ein Verzeichnis namens <dir> im aktuellen Verzeichnis

Jedes Directory beinhaltet die Directories . (aktuelles Directory) und .. (darüberliegendes Directory).

Versteckte Files und Directories beginnen mit einem Punkt (.).

Shortcuts der Shell

Linux / Windows	
Ctrl + C	Beendet das laufende Programm und geht zur Eingabe zurück
TAB	Autovervollständigung (sofern eindeutig)
↑	Holt die vorherige Eingabe in die Prompt zurück
↓	Geht zum nächsten Befehl in der Befehlshistorie

Editor

Zum Erstellen eurer Programme könnt ihr den Editor eurer Wahl verwenden.
Zum Beispiel:

- Für Linux:
 - gedit
 - kate
 - vim
 - emacs
 - etc.
- Für Windows:
 - Notepad++
 - Visual Studio Code
 - Atom (auch für Linux)

Fortran95

- Fortran95 ist eine imperative Programmiersprache, d.h.
„ein Programm besteht aus einer Folge von Anweisungen, die vorgeben, in welcher Reihenfolge was vom Computer getan werden soll“.
- Wie bei einer Fremdsprache, gibt es Vokabeln (hier: Keywords) und eine Grammatik (hier: Syntax).

Unser erstes FORTRAN-Programm

```
PROGRAM hello  
    implicit none  
  
    write(*,*) 'Hello World'  
END PROGRAM
```

Aufgabe

Öffne den Editor deiner Wahl und schreibe dein eigenes Hello-World-Programm. Speichere es in einem entsprechenden Verzeichnis ab.

Einfache Ein- und Ausgabe auf der Konsole

Wenn wir ein Programm über die Kommandozeile öffnen, können wir auch von dieser lesen (genannt Standard Input, kurz: `stdin`) und uns Dinge ausgeben lassen (genannt Standard Output, kurz: `stdout`).

- Eingaben brauchen immer eine Variable, welche mit dem Input beschrieben wird
- Ausgaben überall im Programmablauf möglich
- Bei der Ausgabe von Text die Anführungszeichen (" oder ') nicht vergessen!
- Mehrere Ausgaben mit Kommata (,) voneinander trennen

Einfache Ein- und Ausgabe auf der Konsole

! Die Eingabe über die Konsole (Stdin/Standard Input) wird in

! variable gespeichert

`READ(*,*) variable`

! Mehr als eine Eingabe geht auch

`READ(*,*) variable, other_variable`

! Der Wert von variable wird auf der Konsole (Stdout/Standard

! Output) ausgegeben

`WRITE(*,*) variable`

! Eine Ausgabe mit mehr Inhalt

`WRITE(*,*) "Dieser Text wird so ausgegeben und der Wert von var ist", var`

Aufgabe

- 1 Öffnet den Editor eurer Wahl (z. B. gedit oder kate), erstellt ein neues File und speichert es unter `taschenrechner.f95` ab.
Programmiert einen kleinen Taschenrechner, welcher zwei ganze Zahlen (vom Typ `INTEGER`) von der Konsole einliest und diese addiert. Das Ergebnis soll anschließend auf der Konsole ausgegeben werden.
Testet euer Programm, indem ihr es im Terminal kompiliert und ausführt.

Der GNU Fortran Compiler

Der geschriebene Fortran-Quellcode (source code) eures Programmes wird in einem File <name>.f95 abgespeichert und kann dann mit dem Compiler gfortran in Maschinensprache übersetzt werden. Mittels

```
gfortran taschenrechner.f95
```

wird das Programm übersetzt und ein ausführbares file (executable) namens a.out erstellt.

Um die Executable nach Belieben zu benennen, gibt es den Parameter -o.

```
gfortran -o myAwesomeExecutable taschenrechner.f95
```

Mit dem Parameter -Wall kann man sich beim Kompilieren alle Warnungen ausgeben lassen.

```
gfortran -Wall taschenrechner.f95
```

Der GNU Fortran Compiler

Möchte man sich auf einen Standard beschränken (was eine gute Idee ist), kann man diesen mit dem `-std` parameter angeben:

```
gfortran -std=f95 taschenrechner.f95
```

Wenn man mit Arrays arbeitet (zu einem späteren Zeitpunkt im Semester) und vom Compiler einen Check für zulässige Indexzugriffe haben möchte, ist das mit `-fbounds-check` möglich.

```
gfortran -fbounds-check taschenrechner.f95
```

Diese Parameter sind alle kombinierbar und nachzuschlagen in der `gfortran` Manual. Zum Kompilieren mit Modulen und Bibliotheken kommen wir, wenn wir Module kennenlernen.

Weitere Syntax

Pipe

Die sogenannte Pipe (|) nimmt den Output eines Programms/ Befehls und übergibt diesen an den folgenden Befehl.

Beispiel:

Linux: `fortune | cowsay`

Nützlich ist dies bei eurer ersten Übungsaufgabe. Das Programm erwartet zwei Ganzzahlen als Eingabe. Falls ihr euer Programm mit `roundingerror` bezeichnet habt, dann könnt ihr die Ausführung folgendermaßen abkürzen:

Linux: `echo "665857" "470832" | ./roundingerror`

Windows: `echo "665857" "470832" | roundingerror.exe`

Weitere Syntax

Beispiel (Linux: **grep**, Windows: **find**):

Wir suchen alle Funktionsnamen im File ivalmod.f95.

Linux: `cat ivalmod.f95 | grep -n -i function`

Unabhängig von der Groß- und Kleinschreibung werden alle Zeilen (mit Zeilennummer) ausgegeben, die das Schlüsselwort 'function' enthalten.

Files beschreiben

Output kann in Files geschrieben werden. Mit `>` beschreibt man einen File neu (der alte Inhalt wird überschrieben). Mit `>>` schreib man an das Ende eines Files.

Linux/Windows: `echo "665857470832" | ./roundr > file1`

Linux/Windows: `echo "665857470832" | ./roundr >> file1`

Übungsaufgaben

- Löse Praxisaufgabe 1 des ersten Übungsblattes.
- ZUSATZ: Führe das Programm zur ersten Praxisaufgabe aus, indem du die Eingabewerte direkt übergibst und die Ergebnisse in einer Datei abspeicherst.