

# Einführung

## Sonderübung

Dana Liebscher

PR10

17.10.2023

1. Ablauf
2. Arbeiten mit Linux und Windows
3. Fortran95
  - Unser erstes FORTRAN-Programm
  - Einfache Ein- und Ausgabe auf der Konsole
4. Übungsaufgaben

# Über diesen Kurs

## Vorwissen

- Ihr habt bereits einen Computer benutzt
- Ihr habt hoffentlich bereits einen Compiler und Editor installiert
- Wenn noch nicht: schnellstmöglich machen, bei Fragen gern melden (auch in Konsultation)
- Eventuell habt ihr schon eine (andere) Programmiersprache kennengelernt

## Ablauf

- Insgesamt haben wir ca. 14 Stunden
- Jede Stunde behandelt ein Thema und liefert Übungen, um das neu gewonnene Wissen anzuwenden
- Fragen zu den aktuellen Programmieraufgaben können besprochen werden
- Die Sonderübung dient als Ergänzung zum normalen Übungsbetrieb

# Editor

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

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

Für Notfälle könnt ihr auch Online Fortran Compiler verwenden.

# Einige Quellen

- Ihr könnt uns Tutor:innen fragen
- Opal-Kurs zur Vorlesung  
<https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/41423798272?3>
- Matrix Chat
- 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>
- Leerzeichen zur Abtrennung von Befehlen und Parametern

| Linux                       | Windows                      |   |
|-----------------------------|------------------------------|---|
| <b>man</b><br><befehlsname> | <b>help</b><br><befehlsname> | öffnet das Manual (Handbuch) zum entsprechenden Befehl (schließe mit <b>q</b> für <b>quit</b> ) |
| <b>ls</b>                   | <b>dir</b>                   | listet den Inhalt des Directories auf   |
| <b>cd</b> <directory>       | <b>cd</b> <directory>        | geht in das Directory <directory>   |
| <b>mkdir</b> <directory>    | <b>md</b> <directory>        | erstellt ein Verzeichnis namens <directory> im aktuellen Verzeichnis                            |

# Liste wichtiger Befehle

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

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 helloworld.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 <NameDerDatei.f95> -o <NameSpeicherung>`

```
gfortran helloworld.f95 -o helloworld.exe
```

# 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).
- Beispiele für Keywords:
  - Datentypen(**INTEGER, REAL, CHARACTER**)
  - **PROGRAM, FUNCTION, SUBROUTINE,...**
  - **WRITE, READ**



# 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.

! Die Eingabe über die Konsole wird in variable gespeichert  
`READ(*,*) variable`

! Mehr als eine Variable geht auch  
`READ(*,*) variable1, variable2`

- Ausgaben überall im Programm möglich
- Bei Ausgabe von Text die Anführungszeichen (" oder ') nicht vergessen!
- Mehrere Ausgaben durch Kommata voneinander trennen

! Der Wert von variable wird auf der Konsole ausgegeben

```
WRITE(*,*) variable
```

! Eine Ausgabe mit mehr Inhalt

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

# Aufgabe 1

Erstellt in eurem Editor ein neues File und speichert es unter `taschenrechner.f95` ab.

Programmiert einen kleinen Taschenrechner, welcher zwei ganze Zahlen (also 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.

## Aufgabe 2

Schreibt ein Programm, welches die Quadratzahl zu einer Zahl berechnet. Lest dazu zuerst die gewünschte Zahl von der Konsole ein, berechnet dann die Quadratzahl und gibt das Ergebnis kommentiert aus.

## Aufgabe 3

Schreibe ein Programm, welches den Benutzer nach seinem Namen fragt und ihn begrüßt. Lege dazu eine CHARACTER-Variable an, in der du den Namen speicherst und gib anschließend "Hallo <name>" aus.