Ausgabe von Daten in eine Datei

1. Mit einem Rahmenprogramm soll eine Tabelle produziert werden, mit der die Rechenleistung

   $$\text{Mflops} = \langle \text{Anzahl der Operationen} \rangle \cdot 10^{-6} / \langle \text{gemessene Zeit} \rangle$$

   der verschiedenen Unterprogramme für Vektoroperationen in Abhängigkeit von der Länge der verarbeiteten Vektoren gegenübergestellt wird. Eine Tabellenzeile sollte z.B. enthalten:

   - die Vektorlänge N
   - die Zahl der Mflop/s für `SCAPR`
   - die Zahl der Mflop/s für `VSAXPY`
   - die Zahl der Mflop/s für `ENORM` $\qquad \|x\| = \sqrt{\sum x_i^2}$
   - (evtl. weitere drei Spalten für `double precision`)

2. Die Vektorlänge N sollte zwischen 100 und 100 000 variieren.
   (z.B. $100, 500, 1000, 5000, 10000, 15000, 20000, 30000, 50000, \ldots$)
   Die Anzahl der Wiederholungen eines Unterprogrammaufrufs (zum Zweck der Zeitmessung) ist sinnvollerweise von N abhängig zu wählen, so dass bei jedem N etwa die gleiche Anzahl Operationen in die Zeitmessung eingeht.
   Die Ausgabe in eine Datei kann zunächst ohne spezielle Formatierung erfolgen, indem eine (fast) beliebige Dateinummer (`nr`> 7) und das Standard-Ausgabeformat ($*$) verwendet wird:

   ```
   nr = 9
   ...
   write (nr,*) N,(mflop(k),k=1,6)   ! schreibt eine Tabellenzeile
   ...
   ```

   Dies erzeugt (unter Linux) eine Datei `'fort.9'`.

3. Die einfachste Variante, den Dateinamen selbst zu bestimmen:

   ```
   nr = 9
   OPEN(nr,FILE='dateiname')
   ...
   ...
   ...
   CLOSE(nr)     ! nach Ausgabe der letzten Zeile
   ...
   ```

   Die `CLOSE`-Anweisung ist nicht notwendig, aber zu empfehlen.

4. Zur grafischen Auswertung der Ergebnisse ist `gnuplot` gut geeignet. Das Programm wird (unabhängig vom Fortran-Programm) von der Kommandozeile gestartet und zeigt nach Eingabe folgender Kommandos die Daten der in der Datei gespeicherten Tabelle grafisch an:

```
set data style lines
set grid
set title "Rechenleistung"
plot \
"dateiname" using 1:2 title "scapr", \
"dateiname" using 1:3 title "vsaxpy", \
"dateiname" using 1:4 title "enorm", \
"dateiname" using 1:5 title "dscapr", \
"dateiname" using 1:6 title "vdaxpy", \
"dateiname" using 1:7 title "dnorm"
```

Werden diese Kommandos in eine Datei `plotcmd` geschrieben, so hat man bei gnuplot nur noch einzugeben:

```
load "plotcmd"
```

5. Als weitere Vergleichsmöglichkeit bietet sich an, die Unterprogramme mit unterschiedlichen Compileroptionen zu übersetzen. Man teste z.B. (durch entsprechende Definition der Variablen `FFLAGS=...` im Makefile)

```
gfortran -g   ...
gfortran -O2  ...
gfortran -O3  ...
gfortran -fexpensive-optimizations  ...
gfortran -funroll-all-loops -fomit-frame-pointer ...
```

6. **Komfortablere Arbeit mit Dateien:**
Der Hauptname `filename` der zu schreibenden Dateien soll vom Nutzer eingegeben werden können. Mit Hilfe von Zeichenkettenoperationen sollen aus diesem Wortstamm zwei Dateinamen erzeugt werden, denn das Programm soll unter dem Namen `filename.dat` eine Datei mit der Rechenzeittabelle und unter `filename.plot` eine Datei mit den dazugehörigen Gnuplot-Kommandos schreiben.
Wenn eine der beiden Dateien schon vorhanden ist (mit `INQUIRE` festzustellen), soll eine Warnung erscheinen, woraufhin der Nutzer zu entscheiden hat, ob die Datei überschrieben wird oder ein anderer Name eingegeben werden soll.

   Bitte beachten, dass Zeichenkettenvariable am Ende mit Leerzeichen aufgefüllt sind. Es ist nicht zu empfehlen, durch Verkettung von Zeichenketten Dateinamen der Art `'name      .dat'` zu produzieren. (Funktion `LEN_TRIM(...)` benutzen)

   Verwendung von `INQUIRE`:

```
CHARACTER*20 filename
LOGICAL exists
...
filename=...
INQUIRE (FILE=filename,EXIST=exists)
IF (exists) THEN
   write(*,*) 'schon vorhanden ...'
   ...
ELSE
   OPEN(nr,FILE=filename)
ENDIF
```