Fortran Kontrollstrukturen

Luis Kornblueh

May 28, 2015

Zuweisung

Eine Zuweisung erfolgt über ein `=`.

```
a = 2 + 3
```

Die linke Seite muss eine Variable sein, während die rechte eine Variable, ein Ausdruck oder eine Funktion sein kann.

Fortran erlaubt für Zuweisungen und arithmetische Operatoren auch Felder. Wenn `a`, `b` und `c` Felder mit gleicher Dimensionierung sind, ist es erlaubt, Zuweisungen der Form

```
a = b + c
```

zu schreiben. Besser ist es allerdings, die Dimensionierung mit anzugeben, z.B.

```
a(:,:,:) = b(:,:,:) + c(:,:,:)
```

damit man direkt sieht, dass Felder verwendet werden. Man kann auch Funktionen elementweise anwenden und eine elementweise Multiplikation jedes Feldelements mit einem Skalar durchführen.

Schleifen

Eine Schleife besteht aus den Schlüsselworten

```
do i = 1, 20
   ...
end do
```

Diese Schleife fängt mit `i = 1` an und erhöht die Variable `i` bei jedem Durchlauf um 1, solange bis sie größer als 20 ist.

Möchte man nicht um 1 erhöhen, kann man nach einem weiteren Komma die Schrittgröße angeben.

```
do i = 1, 23, 2
   ...
end do
```

In diesem Beispiel beginnt die Schleife bei `i = 1` und wird bei jedem Schleifendurchlauf um 2 erhöht. Um die Schleife rückwärts zu durchlaufen, kann man einfach eine negative Zahl einsetzen.

Schreibt man eine Schleife einfach

```
do
   ...
end do
```

erhält man eine unendlich durchlaufende Schleife. Dies ist zunächst nicht sinnvoll. Kann aber mit einer Ausstiegsbedingung (`exit`) verknüpft werden.

Der `exit` Ausdruck wird mit einzeiligen `if` Ausdrücken verwendet und führt zur Beendigung einer Schleife.

```
if ( .not. (counter <= 100) ) exit
```

Diese Bedingung würde zur Beendigung einer Schleife führen und das Programm im Ausdruck nach dem Ende der Schleife fortsetzen.

Ähnlich wird `cycle` verwendet. Die Schleife wird mit der nächsten Laufvariablen erneut durchlaufen.

Weiterhin gibt es noch eine `do while` Schleife

```
do while ( Bedingung )
   ...
end do
```

Solange die Bedingung wahr ist, wird die Schleife weiter durchlaufen.

Einfache Bedingung

Der `if` Ausdruck stellt eine einfache Bedingungsanweisung dar.

```
if (a < b) then
   ...
else
   ...
end if
```

Der `else` Ausdruck kann auch weggelassen werden.

```
if (a == b) then
   ...
```

```
end if
```

Man kann aber auch mehrere Bedingung verwenden.

```
if (a >= b) then
   ...
else if (a > c) then
   ...
else
   ...
end if
```

Anstelle von `then` kann auch eine normale Zuweisung stehen.

```
if (a <= 0 ) a = -a
```

Multiple Choice

Neben den einfachen Bedingungen gibt es auch noch einen *switch* Anweisungsblock. Die Abfragefälle dürfen sich nicht überlappen.

```
select case (var)
case (1)
   call sub1
case (2:4)
   call sub2
case default
   call sub3
end select
```