Struktogramme

Struktogramme beschreiben die Grundstruktur eines Programms ohne auf eine bestimmte Programmiersprache einzugehen. Amhand eines Struktogrammes soll es möglich sein, den beschriebenen Algorithmus in jeder Programmiersprache umzusetzen.

1 Elemente eines Struktogramms

1.1 Einfache Anweisungen

c = a + b

Umfasst die Anweisung den Aufruf eines Unterprogramms, so wird dies mit Strichen an der Seite gekennzeichnet:

	c = f(a) + b	

1.2 Verzweigungen

In vielen Situationen ist es nötig, aufgund eines Kriteriums das Vorgehen zu variieren. Ein Beispiel hierfür wäre eine abschnittsweise definierte Situation:

$$y = \begin{cases} 1 & \text{falls } x > 0 \\ 0 & \text{falls } x \leq 0 \end{cases}$$

In einem Struktogramm wird dies folgendermaßen dargestellt:

x > 0	
ja	nein
y = 1	y = 0
	∅

Auf diese Art und Weise können auch mehrfache Verzweigungen dargestellt werden:

$$y = \begin{cases} 1 & \text{falls } x > 1 \\ 0 & \text{falls } 0 \leq x \leq 1 \\ -1 & \text{falss } x < 0 \end{cases}$$

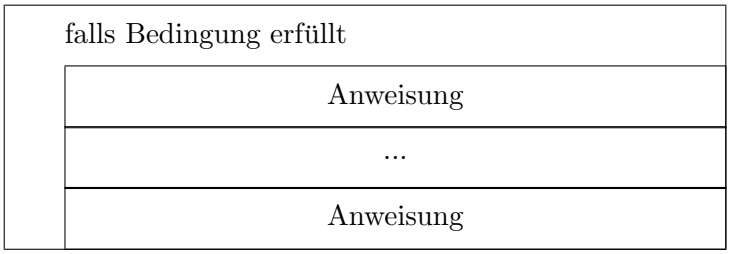
x		
< 0	> 1	sonst
y = -1	y = 1	y = 0

1.3 Wiederholungsschleifen

Wiederholungsschleifen sind das wichtigste Element in einem numerischen Verfahren. In Schleifen werden Anweisungen so lange wiederholt bis ein Abbruchkriterium erfüllt ist. Es gibt verschiedene Arten von Schleifen:

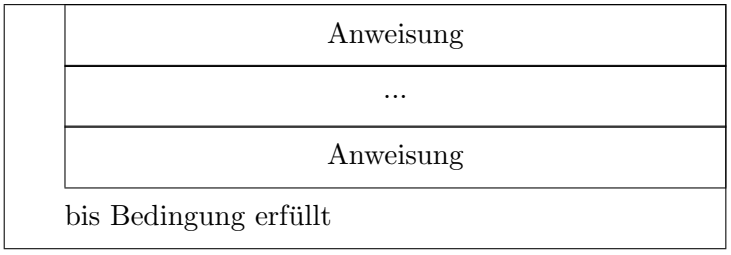
1.3.1 Abweisende Schleifen

Diese Schleifen überprüfen vor einem Durchlauf, ob die Abbruchbedingung erfüllt ist. Ist diese Bedingung nicht erfüllt, werden sie gar nicht erst ausgeführt.



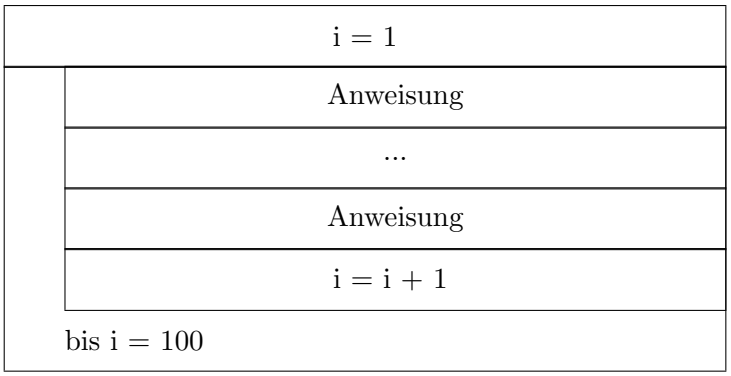
1.3.2 Annehmende Schleifen

Diese Schleifen überprüfen am Ende eines Durchlaufs, ob die Abbruchbedingung erfüllt ist. Dieser Schleifentypus wird immer mindestens einmal ausgeführt, auch dann, wenn die Abbruchbedingung schon vor dem ersten Durchlauf erfüllt wäre.



1.3.3 Zählschleifen

Zählschleifen haben kein eigenes Symbol, da sie durch die anderen beiden Schleifentype dargestellt werden können:



2 Aufbau

Für numerische Verfahren lässt sich der Aufbau eines Struktogramms immer vier Abschnitte einteilen:

1. **Variablendeklaration**

   Hier müssen alle verwendeten Variablen und Funktionen entsprechen ihres Typs (Real oder Integer) deklariert werden.

2. **Initialisierung der Variablen**

   Vor der Verwendung muss jeder Variable ein sinnvoller Wert zugewiesen werden. Dies kann auch durch eine Eingabe vom Benutzer geschehen.

3. **Eigentlicher Algorithmus**

   Hier werden alle Berechnungen durchgeführt

4. **Ausgabe**

   Zum Schluss sollten alle Ergebnisse ausgegeben werden. Dies kann allerdings auch schon in Punkt 3 passieren, falls nötig.

3 Funktionen und Unterprogramme

Unterprogramme sind bei der Entwicklung größerer Codes ein wichtiges Element, um effizient und übersichtlich programmieren zu können. Darüber hinaus ermöglichen Sie es, bereits erstellte Routinen einfach in neue Projekte zu integrieren.

3.1 Funktionen

Funktionen sind unterprogramme, die einen Wert zurückliefern, wie z.B. $y = \sin(x)$. Die Anwendung soll an einem einfachen Beispiel demonstriert werden:

Hauptprogramm:

..
y = quadrat(x)
...

Unterprogramm:

REAL Function quadrat(x)
REAL : x
Übernehme x
quadrat = x * x
Übergebe quadrat

Hervorzuheben ist hier, dass durch „Übernehme“ deutlich gemacht werden soll, welche Variablen tatsächlich aus dem aufrufenden Programm übernommen werden sollen. in „Übergebe“ stehen dann alle Variablen, die wieder an das aufrufenden Programm übergeben werden.

3.2 Unterprogramme

Unterprogramme müssen keinen Rückgabewert haben und werden auch anders aufgerufen. Sie können allgemeine Aufgaben umfassen, die aus dem Hauptprogramm ausgelagert werden sollen. Anzumerken ist, dass es Programmiersprachen wie C gibt, die nur Funktionen kennen. Hier ist es möglich, Funktionen mit „leerem“ Rückgabewert zu definieren, die dann eben „Unterprogramme“ sind. Fortran stellt aber beispielsweise beide Elemente zur Verfügung. Die Verwendung soll an einfachen Beispielen demonstriert werden:

Hauptprogramm:

```
..
    CALL SaveData(x, y, N)
...
```

Unterprogramm:

```
Unterprogramm SaveData(x, y, N)
INTEGER : i, N
REAL : x(1:N), y(1:N)
Übernehme x, y, N
i = 1
Öffne Datei
    Schreibe x(i) und y(i) in Datei
    i = i + 1
bis i = N+1
Schließe Datei
Übergebe nichts
```

Hauptprogramm:

..
CALL GetYZ(x, y, z)
...

Unterprogramm:

Unterprogramm GetYZ(x, y, z)
REAL : x, y, z
Übernehme x
y = x*x
z = 3*x*sqrt(x)
Übergebe y,z