

Einführung in die Informatik I

Kapitel I.5: Verzweigungen

Prof. Dr. Marcin Grzegorzek¹

Research Group for Pattern Recognition
www.pr.informatik.uni-siegen.de

Institute for Vision and Graphics
University of Siegen, Germany



¹Die im Rahmen dieser Lehrveranstaltung verwendeten Lernmaterialien wurden uns zum Großteil von Herrn Prof. Dr. Wolfgang Wiechert und Herrn Prof. Dr. Roland Reichardt zur Verfügung gestellt.

Inhaltsverzeichnis

I. MATLAB-Einführung

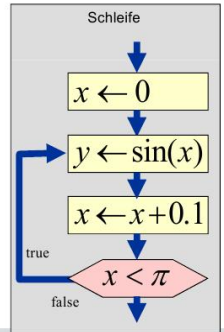
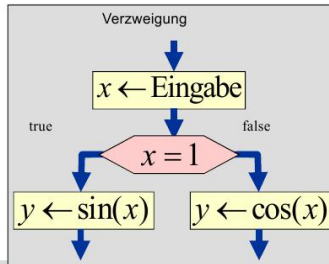
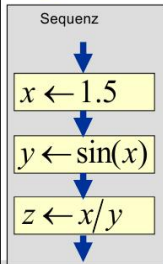
1. Voraussetzungen und Konventionen
2. Variablen und arithmetische Ausdrücke
3. Automatisierungen von Berechnungen
4. Logische Ausdrücke
- ▶ 5. Verzweigungen
6. Schleifen
7. Fehlersuche in Programmen
8. Funktionen
9. Arbeitsweise von Funktionen
10. Vektoren
11. Matrizen

II. Algorithmen

1. Suchen
2. Spezielle Suchalgorithmen
3. Sortieren
4. Rekursion und Quicksort

Die wichtigsten Kontrollstrukturen

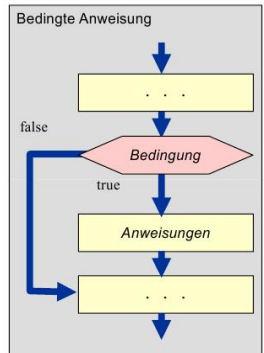
- Bisher besteht ein Programm aus einer festen **Sequenz** von Anweisungen: Der Programmablauf ist immer gleich
- Kontrollstrukturen in Programmiersprachen steuern den Programmablauf in Abhängigkeit von aktuellen Variablenwerten
- Es gibt im wesentlichen 2 Typen davon:
 - **Verzweigungen** und
 - **Schleifen**



Verzweigungen

- Eine Verzweigung – oder auch bedingter Anweisungsblock – hat in MATLAB die Form:

```
if Bedingung  
    Anweisungssequenz  
end
```
- Die Bedingung muss ein boolescher Ausdruck sein:
Ergebnis ist ein Wahrheitswert
- Die Anweisungssequenz wird nur dann ausgeführt, wenn die *Bedingung* wahr (d.h. 1) ist.
- Sonst geht es nach dem **end** weiter.



Beispiel: Stückweise definierte Funktion

- Mathematische Notation

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

- Implementierung mit Hilfe von Verzweigungen

```
if -1<=x & x<0
```

```
  y= x+1;
```

```
end;
```

```
if 0<=x & x<1
```

```
  y=-x+1;
```

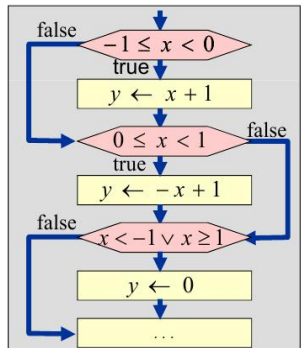
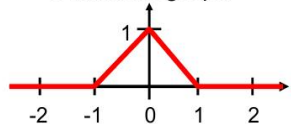
```
end;
```

```
if x<-1 | x>=1
```

```
  y=0;
```

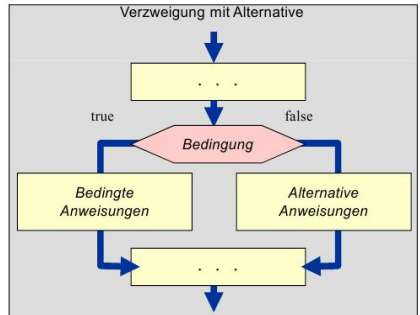
```
end
```

Funktionsgraph



Verzweigung mit Alternative

- Eine Verzweigung mit Alternative hat in MATLAB die Form:
 - **if** *Bedingung*
Bedingte Sequenz
 - else**
Alternative Sequenz
 - end**
- Die alternative Sequenz wird genau dann ausgeführt, wenn die Bedingung falsch (d.h. 0) ist.
- In jedem Fall geht es dann nach dem **end** weiter.

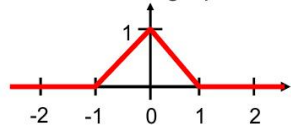


Beispiel: Stückweise definierte Funktion

- Mathematische Notation

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

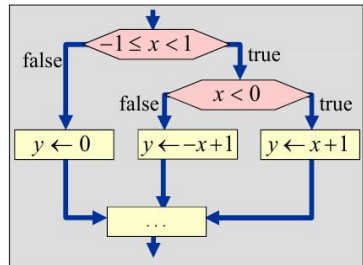
Funktionsgraph



- Implementierung:

```
if -1<=x & x<1
    if x<0
        y= x+1;
    else
        y=-x+1;
    end
else
    y=0;
end
```

Blockstruktur wird durch
Einrücken deutlich



MATLAB unterstützt
automatisches Einrücken

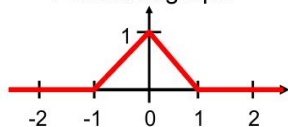


Beispiel: Stückweise definierte Funktion

- Mathematische Notation

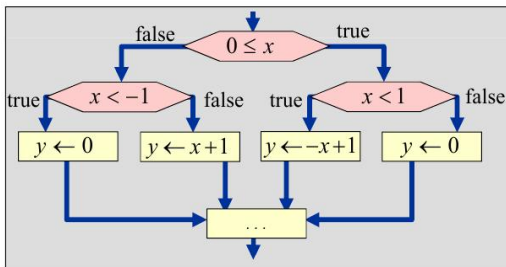
$$y = \begin{cases} x+1 & \text{falls } -1 \leq x < 0 \\ -x+1 & \text{falls } 0 \leq x < 1 \\ 0 & \text{sonst} \end{cases}$$

Funktionsgraph



- Alternative Implementierung:

```
if 0 <= x
  if x < 1
    y = -x + 1;
  else
    y = 0;
  end
else
  if x < -1
    y = 0;
  else
    y = x + 1;
  end
end
```



Fallunterscheidungen

- Eine Fallunterscheidung hat in MATLAB die Form:

```
□ if      Fall 1
      Sequenz 1
elseif Fall 2
      Sequenz 2
elseif Fall 3
      Sequenz 3
      . . .
else
      alternative Sequenz
end
```

- Es darf immer nur einer der Fälle oder die Alternative eintreten.
- Der `else`-Zweig kann wegfallen.
- Flußdiagramme kennen keine Notation für Fallunterscheidungen

- Die Fallunterscheidung ist eine Abkürzung für:

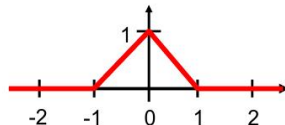
```
□ if Fall 1
      Sequenz 1
else
      if Fall 2
          Sequenz 2
      else
          if Fall 3
              Sequenz 3
          else
              Alternat.
              Sequenz
          end
      end
end
```

Beispiel: Stückweise definierte Funktion

■ Mathematische Notation

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

Funktionsgraph



■ Implementierung mit einer Fallunterscheidung:

```
if x < -1
    y = 0;
elseif -1 <= x & x < 0
    y = x + 1;
elseif 0 <= x & x < 1
    y = -x + 1;
else
    y = 0;
end;
```

Programmierstil

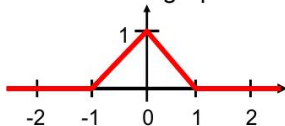
- Viele Programmvarianten können dasselbe Problem lösen
- Die Problemlösung ist mehr oder weniger übersichtlich
- Ohne sorgfältiges Einrücken der Blöcke wird ein Programm schnell unlesbar
- Flussdiagramme werden schnell sehr raumgreifend

Mehrere zutreffende Alternativen

- Es können mehrere Alternativen zutreffen:

```
if      x < -1
  y = 0;
elseif  x < 0
  y = x + 1;
elseif  x < 1
  y = -x + 1;
else
  y = 0;
end;
```

Funktionsgraph

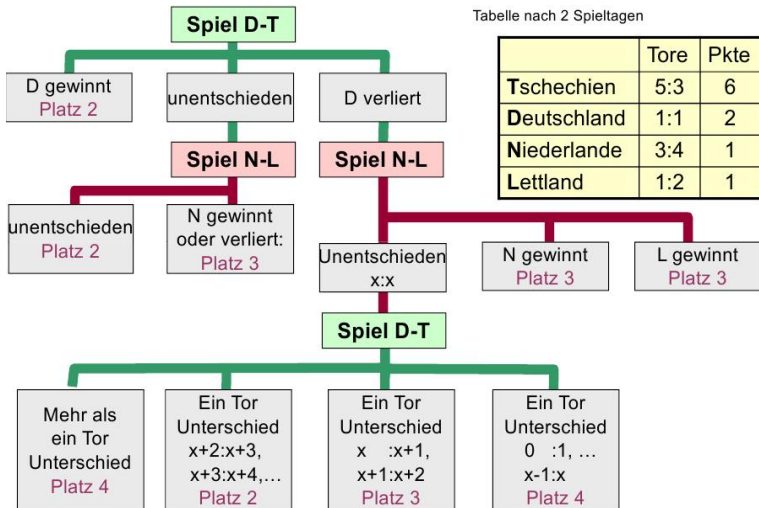


- Mathematische Notation

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

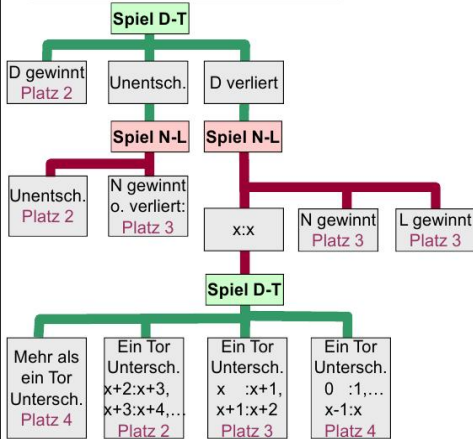
- Trotzdem wird nur **eine** Alternative ausgeführt:
 - $x = -1.5 \rightarrow y = 0.0$
 - $x = 0.5 \rightarrow y = 0.5$
 - $x = -0.5 \rightarrow y = 0.5$
 - $x = 1.5 \rightarrow y = 0.0$

Beispiel: EM-Vorrunde 2004



Pseudocode für das EM-Beispiel

Eingabe: T, D, N, L -- erzielte Tore
Ausgabe: P -- Platz D



```

if  $D > T$  --  $D$  gewinnt
   $P \leftarrow 2$ ;
elseif  $D = T$  --  $D$  unentsch.
  if  $N = L$ 
     $P \leftarrow 2$ ;
  else
     $P \leftarrow 3$ ;
  endif;
else --  $D$  verliert
  if  $N \neq L$  --  $N$  oder  $L$  gew.
     $P \leftarrow 3$ ;
  else --  $N - L$  unentsch.
     $x \leftarrow N$ ;
    if  $T > D + 1$ 
       $P \leftarrow 4$ ;
    elseif  $D \geq x + 2$ 
       $P \leftarrow 2$ ;
    elseif  $D \geq x$ 
       $P \leftarrow 3$ ;
    else
       $P \leftarrow 4$ ;
    endif;
  endif;
endif;
  
```

Pseudocode im Vergleich zu MATLAB

```
if    D > T    -- D gewinnt
    P ← 2;
elseif D = T  -- D unentsch.
    if N = L
        P ← 2;
    else
        P ← 3;
    endif;
else   -- D verliert
    if N ≠ L  -- N oder L gew.
        P ← 3;
    else     -- N - L unentsch.
        x ← N;
        if T > D + 1
            P ← 4;
        elseif D ≥ x + 2
            P ← 2;
        elseif D ≥ x
            P ← 3
        else
            P ← 4
        endif;
    endif;
endif;
```



```
T = input('Tore Tschechien : ');
D = input('Tore Deutschland : ');
N = input('Tore Niederlande : ');
L = input('Tore Lettland : ');

if    D > T    % D gewinnt
    P = 2;
elseif D == T % D unentschieden
    if N == L
        P = 2;
    else
        P = 3;
    end;
else % D verliert
    if N ~ L
        P = 3;
    else
        x = N;
        if T > D + 1
            P = 4;
        elseif D >= x + 2
            P = 2;
        elseif D >= x
            P = 3
        else
            P = 4
        end;
    end;
end;
disp('Platz Deutschland'); disp(P);
```