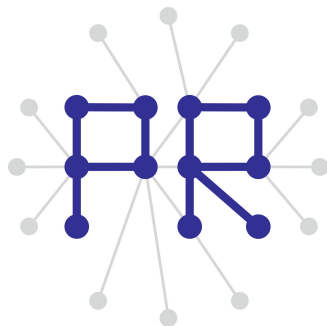
Einführung in die Informatik I

Kapitel I.8: Funktionen

Prof. Dr. Marcin Grzegorzek[1]

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

Institute for Vision and Graphics
University of Siegen, Germany



[1] Die 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

Inhaltsverzeichnis II

Funktionen

- Funktionen (auch Prozeduren, Unterprogramme) sind ein weiteres programmiersprachliches Hilfsmittel, um den Programmablauf zu steuern
- Mit Hilfe von Funktionen können Rechenvorschriften auf eine wiederverwendbare Art verfügbar gemacht werden
- Eine solche Vorschrift f berechnet m Ausgaben (Ergebnisse) auf der Grundlage von n Eingaben:

$$f: (x_1, \ldots, x_n) \rightarrow (y_1, \ldots y_m)$$

- Funktionen werden in Bibliotheken gesammelt
- Die MATLAB-Funktionsbibliothek stellt bereits eine Fülle von Funktionen bereit:
  - `x   =sin(y);    % 1 Eingabe , 1 Ausgabe`
  - `Rest=rem(20,3);% 2 Eingaben, 1 Ausgabe`
  - `plot(1,1);      % 2 Eingaben, 0 Ausgaben`
  - `[Z,N]=rat(x);   % 1 Eingabe , 2 Ausgaben`

Wiederholung über Skripte

- Wiederverwendung einer Berechnungsvorschrift mit den bisher bekannten Mitteln (Skripte)

Skript A1

```
% Skript zum wiederholten
% Aufruf von Skript A2

x=0;

while x<=1
    SkriptA2;
    disp(y);
    x=x+0.1;
end;
```

Skript A2

```
% Skript zur Berechnung
% der Funktion y=sin(2*pi*x)
% Eingangsvariable: x
% Ausgangsvariable: y



y=sin(2*pi*x);
```

- Auch komplexere Berechnungen können so mehrfach wiederverwendet werden.

Probleme mit Skripten

- Allerdings tritt bei dieser Methode ein schwerwiegendes Problem auf:

Skript B1

```
% Skript zum wiederholten
% Aufruf von Skript B2

x=0;
c=0.1;
while x<=1
    SkriptB2;
    disp(y);
    x=x+c;
end;
```

Skript B2

```
% Skript zur Berechnung der
% Funktion y=sin(2*pi*x)
% Eingangsvariable: x
% Ausgangsvariable: y
% Hilfsvariable   : c

c=2*pi;
y=sin(c*x);
```

Wertüberschreibung

- Das Konzept produziert Fehler, wenn aufrufendes und aufgerufenes Skript gleiche Variablen verwenden.
- In der Praxis is dieses Vorgehen daher nicht praktikabel.

Funktionen lösen das Problem

- Eine Funktion versteckt die in ihr eingeführten Variablen vor dem Zugriff aus dem aufrufenden Skript:

Skript C1

```
% Skript zum wiederholten
% Aufruf von Skript C2

x=0;
c=0.1;
while x<=1
    y=SkriptC2(x);
    disp(y);
    x=x+c;
end;
```

Skript C2

```
function y=SkriptC2(x);
% Funktion zur Berechnung
% von y=sin(2*pi*x)
% Eingangsvariable: x
% Ausgangsvariable: y
% Hilfsvariable   : c

c=2*pi;
y=sin(c*x);
```

c ist eine lokale Variable

- Die in einer Funktion eingeführten Variablen heißen lokale Variablen. Für diese wird neuer Speicherplatz angelegt.

Übergabeparameter von Funktionen

- Die Übergabevariablen und Rückgabevariablen einer Funktion müssen nicht dieselben Namen haben wie im Aufruf. Sie sind ebenfalls lokale Variablen:

Skript D1

```
% Skript zum wiederholten
% Aufruf von Skript D2

a=0;
c=0.1;
while a<=1
    b=SkriptD2(a);
    disp(b);
    a=a+c;
end;
```

Skript D2

```
function y=SkriptD2(x);
% Funktion zur Berechnung
% von y=sin(2*pi*x)
% Eingangsvariable: x
% Ausgangsvariable: y
% Hilfsvariable   : c

c=2*pi;
y=sin(c*x);
```

- Beim Aufruf werden die Variablenwerte aus dem Aufruf für die Parameter der Funktion eingesetzt. Hier: a→x
- Ebenso bei der Rückgabe des Ergebnisses. Hier: y→b

Wiederholung von Programmteilen

- Beispiel: Parametereingabe bei der Zinsrechnung

```
Kredit    = input('Kredit '                    );
Prozente = input('Prozentsatz pro Monat ');
Monatl    = input('Monatlicher Betrag '    );
```

- Bei unsinnigen Eingaben werden falsche Ausgaben berechnet:
  - Der Kredit muss mindestens 1000 betragen. Andererseits gibt es eine Obergrenze, z.B. 1000000.
  - Die Prozentzahl muss im Bereich von 0.001 bis 0.005 liegen.
  - Die Monatliche Abzahlung muss mindestens die Zinsen abtragen. Sie darf aber 1/100 des Kredits nicht überschreiten.
- Falsche Eingaben sollen im Programm abgefangen werden:

```
Kredit = input('Kredit');
while (Kredit <= 1E3) | (Kredit > 1E6)
    disp('Falsche Eingabe');
    Kredit = input('Neue Eingabe ,);
end
```

Wiederholung von Programmteilen

- Vollständige Korrektheitsabfragen:

```
Kredit = input('Kredit');
while (Kredit <= 1E3) | (Kredit > 1E6)
    disp('Falsche Eingabe');
    Kredit = input('Neue Eingabe ');
end
```

```
Prozente = input('Prozente');
while (Prozente <= 0.001) | (Prozente > 0.005)
    disp('Falsche Eingabe');
    Prozente = input('Neue Eingabe ');
end
```

```
Monatl = input('Monatl');
while (Monatl <= Prozent*Kredit)...
                    | (Monatl > Kredit/10)
    disp('Falsche Eingabe');
    Monatl = input('Neue Eingabe ');
end
```

immer gleicher Code

beschreibender Text

erlaubte Untergrenze

erlaubte Obergrenze

einzugebender Wert

Texte in MATLAB

- Texte werden in der Programmierung auch als Zeichenketten oder Strings bezeichnet
- In MATLAB stehen Texte in Hochkommata:
  - `'Kredit'`
  - `'Falsche Eingabe'`
- Strings können wie Gleitkommazahlen in Variablen gespeichert werden:
  - `Text   = 'Kredit'`
  - `Prompt ='Eingabe: '`
- Die **`disp`**- und **`input`**-Befehle geben Texte ohne die Hochkommata aus:
  - `disp(Text)` → **`Kredit`**
  - `x=input(Prompt)` → **`Eingabe:`**
- Der Workspace-Browser zeigt Textvariablen entsprechend an.

Implementierung als Funktion

- Einsetzen von Platzhaltern mit Namen:

```
Wert = input( Text );
while ( Wert <= Unter ) | ( Wert > Ober )
    disp('Falsche Eingabe');
    Wert = input('Neue Eingabe ');
end
```

- Kapseln als Funktion:

```
function Wert = CheckedInput ( Text, Unter, Ober )
% Geprüfte Eingabe eines Wertes . . .
Wert = input( Text );
while ( Wert <= Unter ) | ( Wert > Ober )
    disp('Falsche Eingabe');
    Wert = input('Neue Eingabe ');
end
```

- Anwendung der Funktion:

```
K=CheckedInput('Kredit           ',1E3   ,1E6     );
P=CheckedInput('Prozents./Monat',0.001,0.005  );
M=CheckedInput('Monatl. Betrag',Prozent*Kredit,Kredit/10)
```

Nachträgliche Erweiterung

- Ändert man eine Funktionsimplementierung, so ändert sich ihr Verhalten überall auf die gleiche Weise
- Beispiel:

```
function Wert = CheckedInput ( Text, Unter, Ober )
% Geprüfte Eingabe eines Wertes . . .
Wert = input( Text );
while ( Wert <= Unter ) | ( Wert > Ober )
    if Wert <= Unter
        disp('Wert muss größer als ');
        disp(Unter); disp(' sein.');
    else
        disp('Wert muss kleiner als ');
        disp(Ober); disp(' sein.');
    end
    Wert  = input('Neue Eingabe ');
end
```

- Der Code der Anwendung muss nicht verändert werden:

```
K=CheckedInput('Kredit           ',1E3,1E6      );
P=CheckedInput('Prozents./Monat',0.1,0.1      );
M=CheckedInput('Monatl. Betrag ',0.0,Kredit/10);
```

Übergabe-Parameter

- Funktionen können 0,1,2,… Variablen als Eingabewert haben.
- Grundsätzlich muss für jeden Eingabeparameter im Aufruf auch ein Wert bereitgestellt werden. Sonst gibt es eine Fehlermeldung.
- Beispiel:
  - `Kredit =CheckedInput('Kredit ');`
    → `??? Input argument "Unter" is undefined.`
  - `Kredit =CheckedInput('Kredit ',0);`
    → `??? Input argument „Ober" is undefined.`
- Der Fehler tritt erst dann auf, wenn die entsprechende Variable im Funktionsrumpf gebraucht wird.

Mehrere Rückgabewerte

- Funktionen können in MATLAB 0,1,2,… Werte zurückgeben.
- Beispiel: Zinsrechnung
  - Eingabe: `Kredit, Prozente, Monatl_Betrag`
  - Ausgabe: `Laufzeit, Gesamtkosten, Letzte_Rate`
- Implementierung:

```
function [Laufzeit,Gesamtkosten,Letzte_Rate]...
         = ZinsRechnung(Kredit,Prozente,Monatl_Betrag)
Laufzeit     = 0;
Gesamtkosten = 0;
while Kredit > 0
    Laufzeit = Laufzeit + 1;
    Tilgung  = Monatl_Betrag - Prozente * Kredit ;
if Tilgung >= Kredit % Kreditende erreicht
        Tilgung       = Kredit;
        Letzte_Rate   = Prozente * Kredit + Tilgung;
        Gesamtkosten  = Gesamtkosten + Letzte_Rate;
        Kredit = 0;
    else % Kreditende noch nicht erreicht
        Kredit        = Kredit - Tilgung;
        Gesamtkosten  = Gesamtkosten + Monatl_Betrag;
    end
end
```

Hilfsvariable

Funktion mit mehreren Ergebnissen

- Funktionen mit mehreren Ergebnissen haben die Form:

```
function [Resultat1, Resultat2,...]
  = Funktionsname (Parameter1, Parameter2,...)
Funktionsrumpf
```

- Aufrufendes Skript:

```
K=CheckedInput('Kredit ',1E3,1E6    );
P=CheckedInput('Prozent',0.001,0.005);
M=CheckedInput('Monatl.',P*K,K/10   );
[LZ,GK,LR]=ZinsRechnung(K,P,M);
```

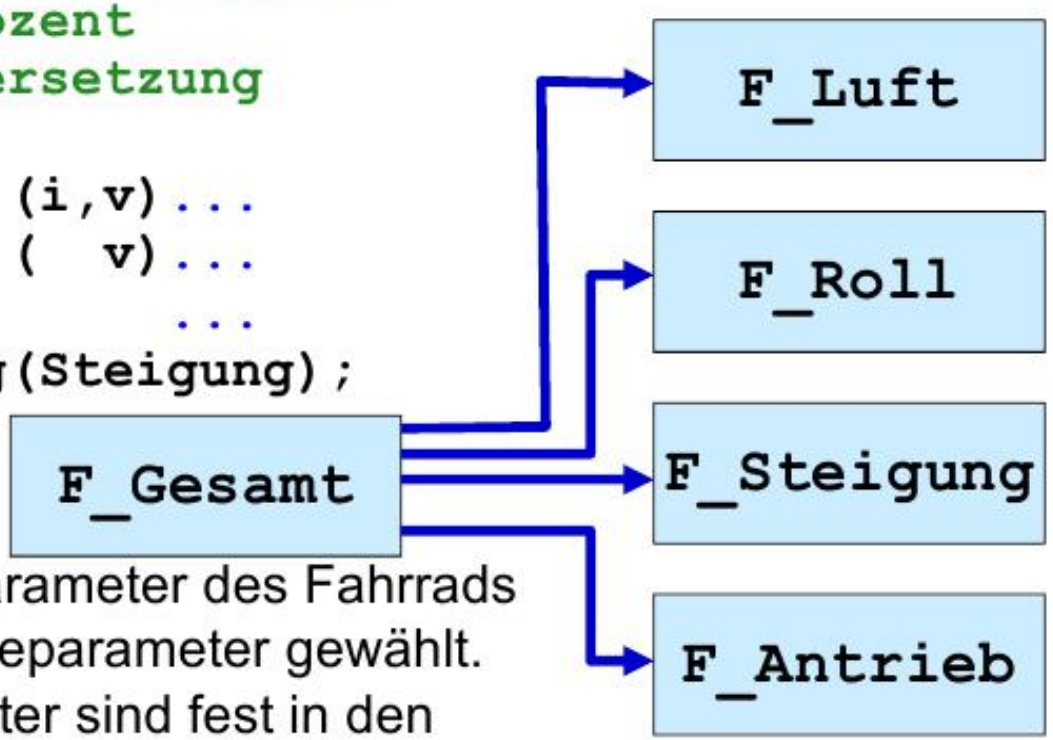
- Alternative Aufrufe mit 1-3 abgerufenen Ergebnissen:_

```
[LZ,GK,LR]=KreditRechnung(1000,0.04,100)
[LZ,GK]   =KreditRechnung(1000,0.04,100)
 LZ       =KreditRechnung(1000,0.04,100)
```

Funktionen rufen andere Funktionen auf

```
function F=F_ges(v, Steigung, i)
% Alle Gegenkäfte, die auf
% den Fahrradfahrer wirken
% v         - Geschwindigkeit
% Steigung - Prozent
% i         - Übersetzung

F =    F_Antrieb (i,v)...
     - F_Luft    (  v)...
     - F_Roll          ...
     - F_Steigung(Steigung);
```



- Nur ausgewählte Parameter des Fahrrads werden als Übergabeparameter gewählt.
- Die übrigen Parameter sind fest in den Funktionen vorgegeben.