Grundlagen der Programmierung

8. Vorlesung
05.12.2017

Kontrollstrukturen

Verzweigungen

```
if bedingung1
  block
elseif bedingung2
  block
elseif bedingung3
  block
...
else
  block
end
```

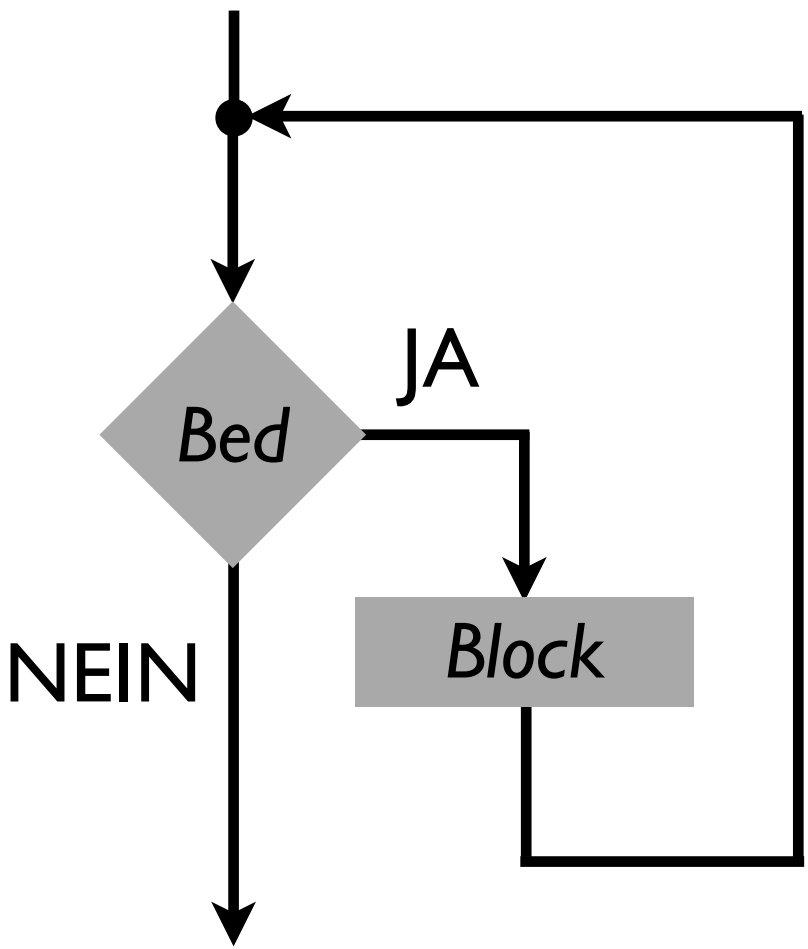
elseif...
else...
kann wegfallen

```
switch switch_ausdruck
  case case_ausdruck1
      Anweisungen;
  case case_ausdruck2
      Anweisungen;
  ...
  otherwise
      Anweisungen;
end
```

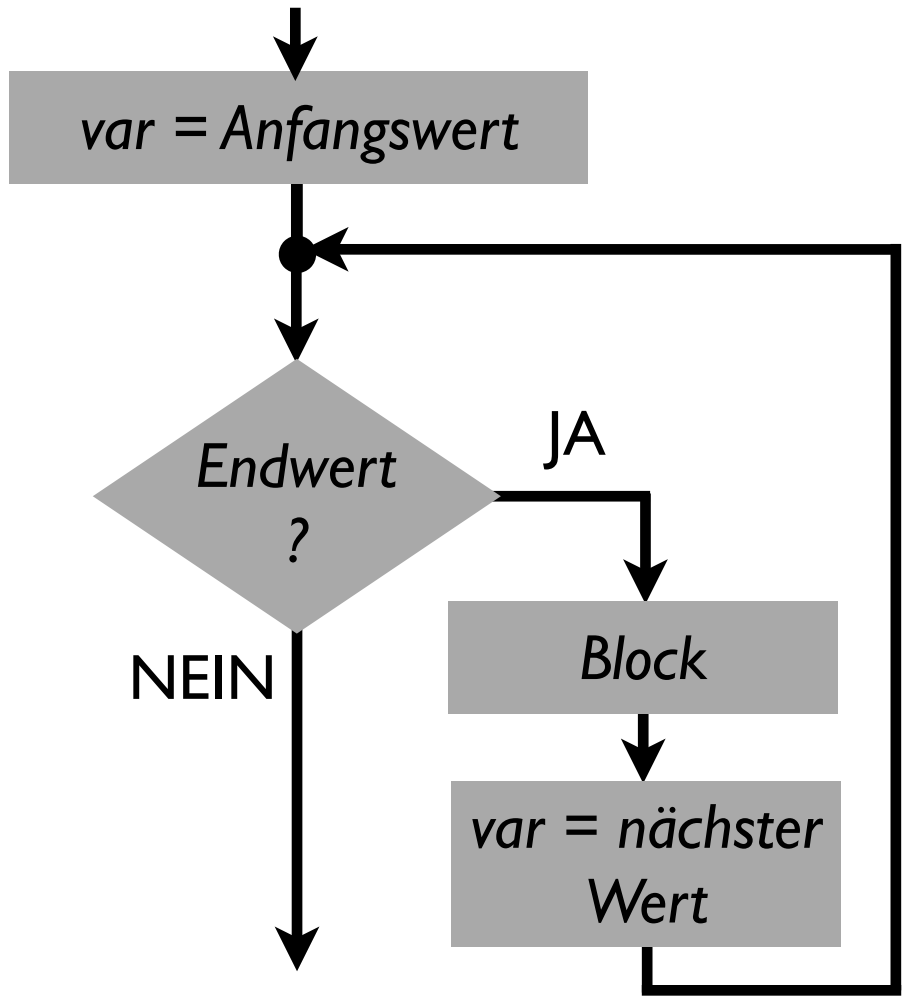
switch_ausdruck:
Skalar o. Zeichenkette
case_ausdruck:
Konstante o. Skalar o. Array { *el1*, *el2*, ... }

Schleifen

```
while bedingung
   block
end
```




```
for var = bereich
   block
end
```



Kontrollstrukturen: `for`

- Unbedingte Scheife
- ```
  for variable = bereich
      Block
  end
  ```
- Steuerung der Schleife über die Bereichsangabe
- Schleifenvariable nimmt alle Werte im Wertebereich an
- Auch: Iteration über alle Elemente eines Vektors/ einer Matrix
  ```

Bereichsangaben

```
for n = 1:10
  fprintf('%d', n);
end
```

$n = 1,2,3,4,5,6,7,8,9,10$

```
for n = 1:2:10
  fprintf('%d', n);
end
```

$n = 1,3,5,7,9$

```
for n = 10:-1:0
  fprintf('%d', n);
end
```

$n = 10, 9, 8, \ldots, 0$

```
for n = [4 8 15 16 23 42]
  fprintf('%d', n);
end
```

$n = 4,8,15,16,23,42$

Geschachtelt

Dieser Block wird
N mal ausgeführt...

```
for n = 1:N
    for m = 1:M
        fprintf('(%d,%d)', n, m);
    end
end
```

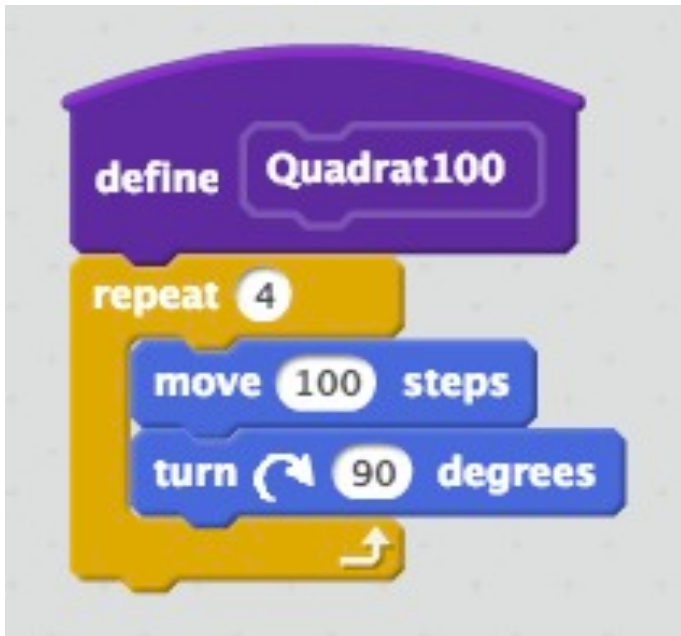
Dieser Block wird
*N***M* mal ausgeführt

n und *m* nehmen alle
Wertepaare (*n*, *m*) mit
n ∈{ 1,2 ... , *N* } und
m ∈{ 1,2 ... , *M* } an

Funktionen

Funktionen

- "neue Befehle" definieren



- Ziel: Programmtext gliedern / strukturieren
- Teilprobleme lösen
- Wiederverwertbaren Code schreiben

Funktionen als Unterprogramme

```
hilfe_ausgeben();
fprintf(“hier geht’s weiter...\n”);


function hilfe_ausgeben
   fprintf("Tolles Programm...");
end
```

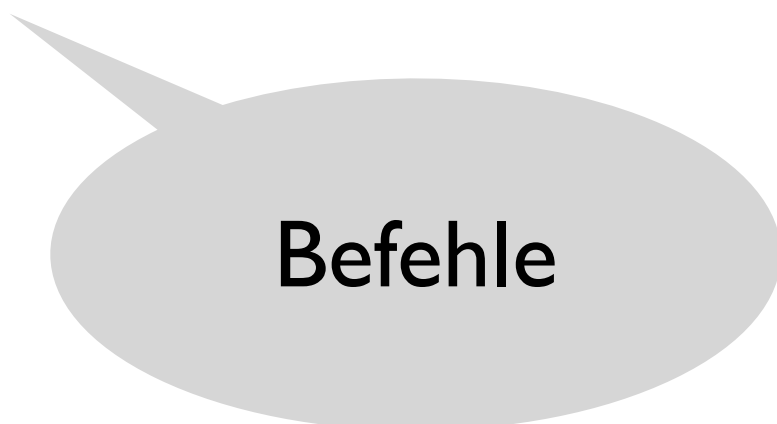
Funktionsdeklaration

Ziel: Programm strukturieren

Funktionen als Unterprogramme

- Allgemeine Form (vereinfacht):

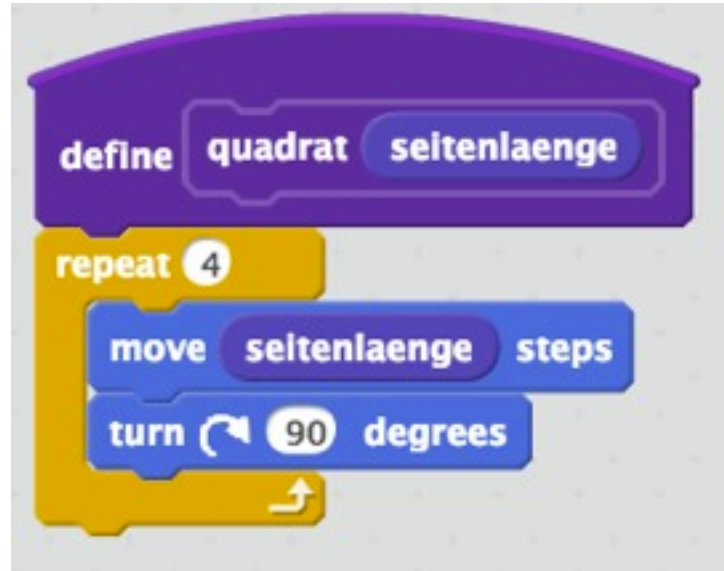
```
function name
   Funktionsrumpf
end
```



- Funktionen unter dem Hauptprogramm definieren

Funktionen mit Parametern

- Funktion, die nicht ohne weitere Angaben arbeiten kann
- Beispiel: wie lang soll das Quadrat werden?
- *seitenlaenge* als Parameter



Funktion mit Parameter: Deklaration

formaler Parameter

```
function print_n_stars( n )
    for x = 1:n
        fprintf('*');
    end
end
```

Formale Parameter:
Angaben, die die Funktion "braucht", um Arbeiten zu können

Funktionen mit Parameter: Aufruf

aktueller Parameter

```
print_n_stars( 5 );


function print_n_stars( n )
    for x = 1:n
        fprintf('*');
    end
end
```

formaler Parameter

Funktion `print_n_stars` wird mit *n*=5 ausgeführt

Funktionen mit Parametern

Die Auswirkung des Aufrufs `print_n_stars(5)` kann man sich wie folgt vorstellen:

```
function print_n_stars()
    n = 5;
    for x = 1:n
        fprintf('*');
    end
end
```

Funktionen mit Parametern

```
n = input('Wie hoch? ');
tannenbaum( n );

function print_n_stars( n )
    for x = 1:n
        fprintf('*');
    end
end

function tannenbaum( n )
    for x = 1:n
        print_n_stars( x );
        fprintf('\n');
    end
end
```

unterschiedliche Variablen!

```
*
**
***
****
```

Formale und aktuelle Parameter

```
irgendeine(5,'x',7);
```

Aktuelle Parameter oder *Argumente*

```
function irgendeine( a, b, c )
    fprintf('a= %d\n', a);
    fprintf('b= %c\n', b);
    fprintf('c= %d\n', c);
end
```

Formale Parameter

Beim Aufruf der Funktion werden die *formalen Parameter* mit den Werten der *aktuellen Parameter* belegt.

Aufgabe: Wortgenerator

- Präfixe: Haus, Land, Meer, Flug, Borsten, Kampf
- Tiere: Schwein, Hund, Katze, Elefant, Maus, Goldfisch, Saurier
- Gib alle Kombinationen aus Präfixen und Tieren aus
- Hausschwein, Landschwein, Meerschwein, Flugschwein, Borstenschwein, Kampfschwein, Haushund, Landhund, Meerhund, Flughund, Borstenhund, Kampfhund, ...

1. Teilaufgabe

- Funktion "Wort ausgeben"
- Eingabe: Integer-Zahl *i*
- Die Funktion soll das *i*. Wort der Wortliste ausgeben
  - 1: Haus
  - 2: Land
  - 3: Meer usw.

Teilaufgabe I

```
function printName( n )
  switch n
    case 1
        fprintf('Haus');
    case 2
        fprintf('Land');
    case 3
        fprintf('Meer');
...
    case 12
        fprintf('goldfisch');
    case 13
        fprintf('saurier');
    otherwise
        fprintf('-');
  end
end
```

2. Teilaufgabe

- Konstruiere Schleifen, die mit Hilfe von Teilaufgabe 1 alle Wortkombinationen ausgeben
- Erste Schleife: Alle Tiere
- Zweite Schleife: Alle Präfixe
- Variablen (Konstanten):

```
max_praefix = 6;
max_tier = 13;
```

2. Teilaufgabe

```
max_praefix = 6;
max_tier = 13;

for t = max_praefix + 1:max_tier
    for p = 1:max_praefix
        printName( p );
        printName( t );
        fprintf('\n');
    end
end
```