Grundlagen der Programmierung

7.Vorlesung
28.11.2017

Kontrollstruktur: if ... else

- Allgemein:

```
if bedingung
  block
else
  block
end
```

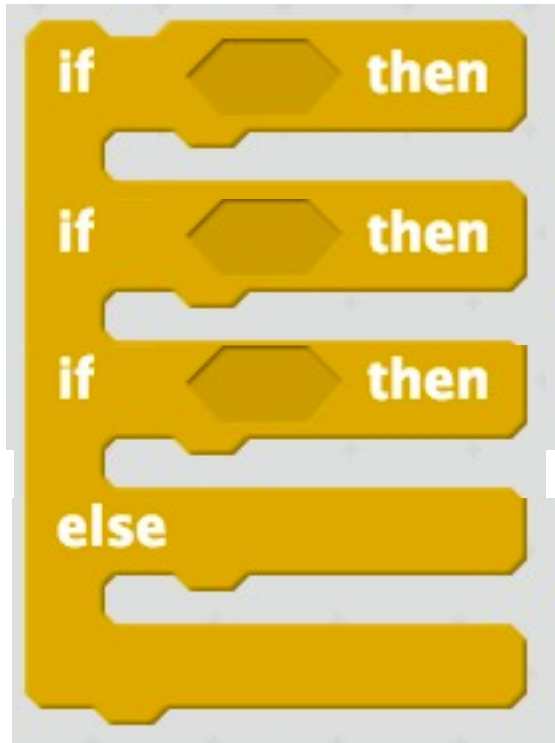
- else-Zweig kann entfallen

```
if bedingung
  block
end
```

Kontrollstruktur: if ... elseif ... ... else

- geschachtelt:

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



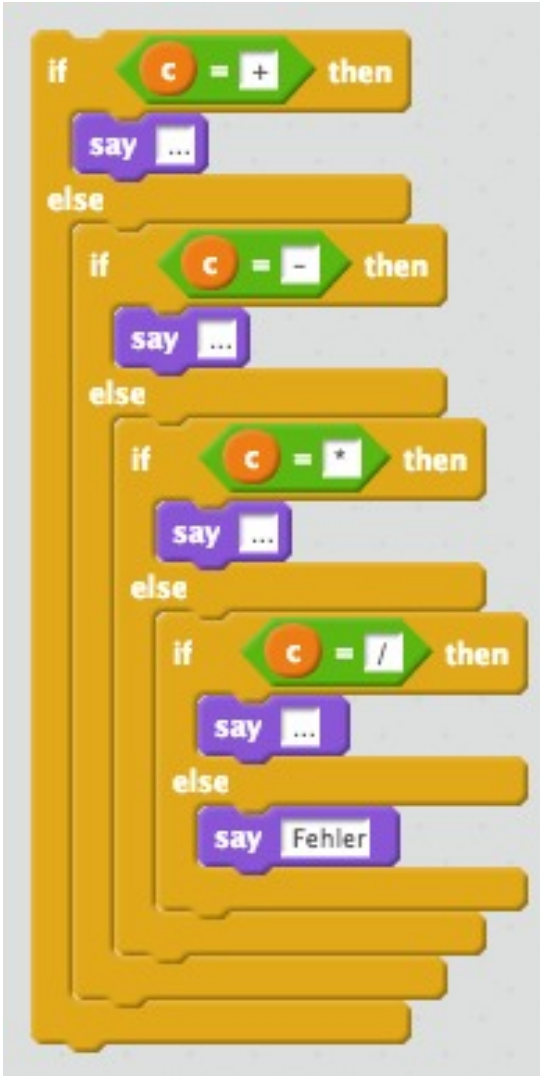
Aufgabe: Mini-Taschenrechner

- Lies eine float-Zahl, einen Operator (+, -, * oder / als char) und eine weitere float-Zahl von Tastatur ein.
- Gib das Ergebnis des Operator angewendet auf die beiden Operanden aus.
- Gib eine Fehlermeldung aus, wenn ein ungültiger Operator eingegeben wurde.

Aufgabe: Mini-Taschenrechner

```
op1 = input('Erster Operand: ');
c = input('Operation: ', 's');
op2 = input('Zweiter Operand: ');

if c == '+'
    fprintf( ‘%f’,  op1 + op2 );
elseif c == '-'
    fprintf( ‘%f’,  op1 - op2 );
elseif c == '*'
    fprintf( ‘%f’,  op1 * op2 );
elseif c == '/'
    fprintf( ‘%f’,  op1 / op2 );
else
    fprintf( ‘Fehler: Falscher Op.!\n’ );
end
```

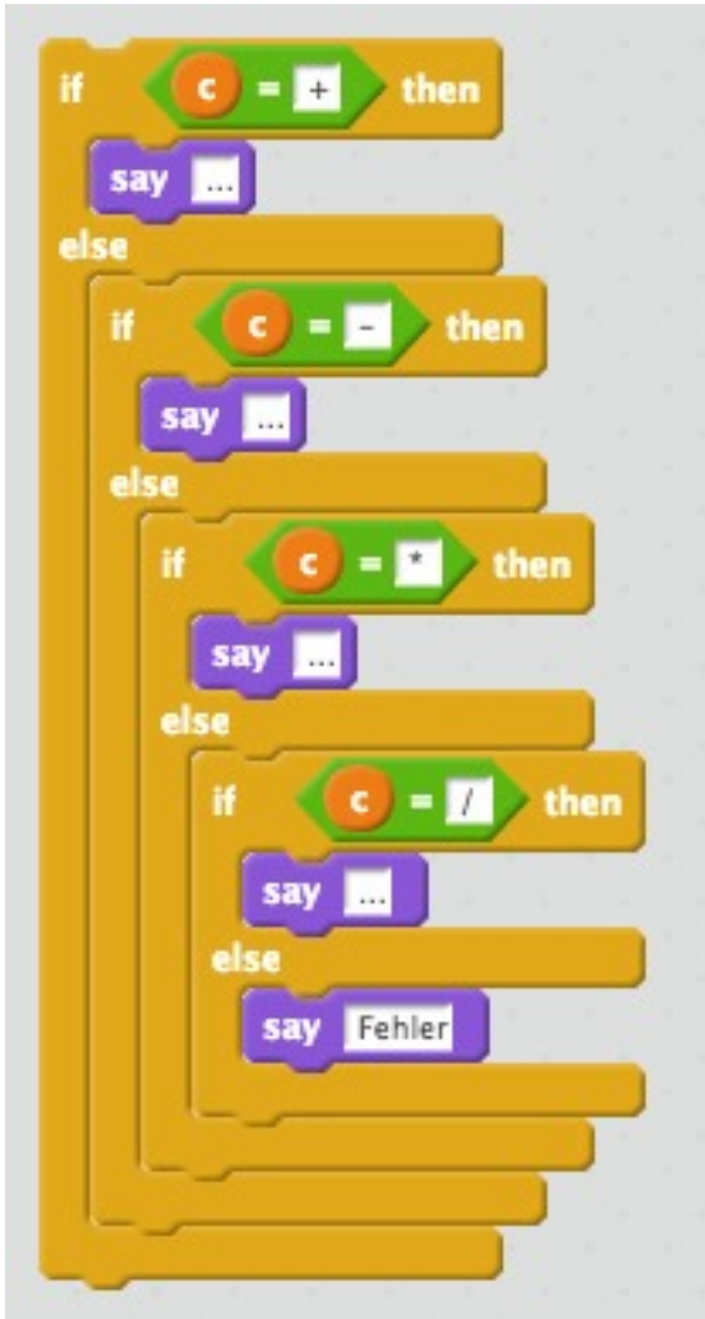


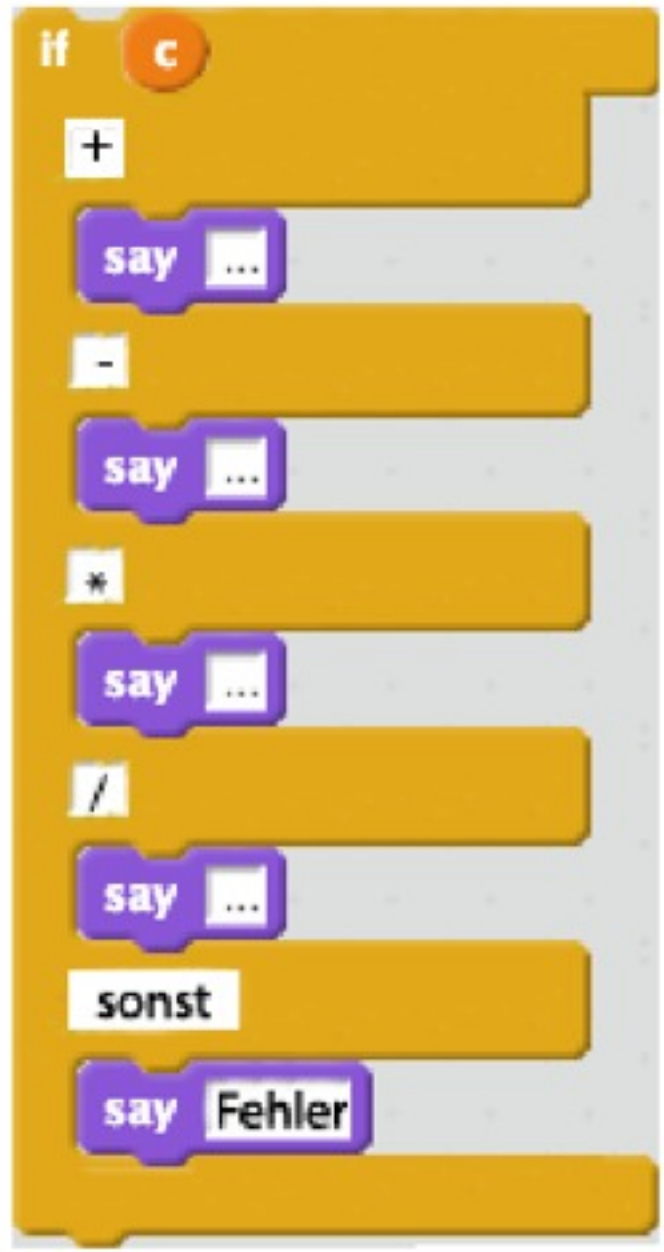
Aufgabe: Mini-Taschenrechner

```
op1 = input('Erster Operand: ');
c = input('Operation: ', 's');
op2 = input('Zweiter Operand: ');

switch c
    case '+'
      fprintf( '%f',  op1 + op2 );
    case '-'
      fprintf( '%f',  op1 - op2 );
    case '*'
      fprintf( '%f',  op1 * op2 );
    case '/'
      fprintf( '%f',  op1 / op2 );
    otherwise
      fprintf( 'Fehler: Falscher Op.!\n' );
end
```

Mehrfachverzweigung

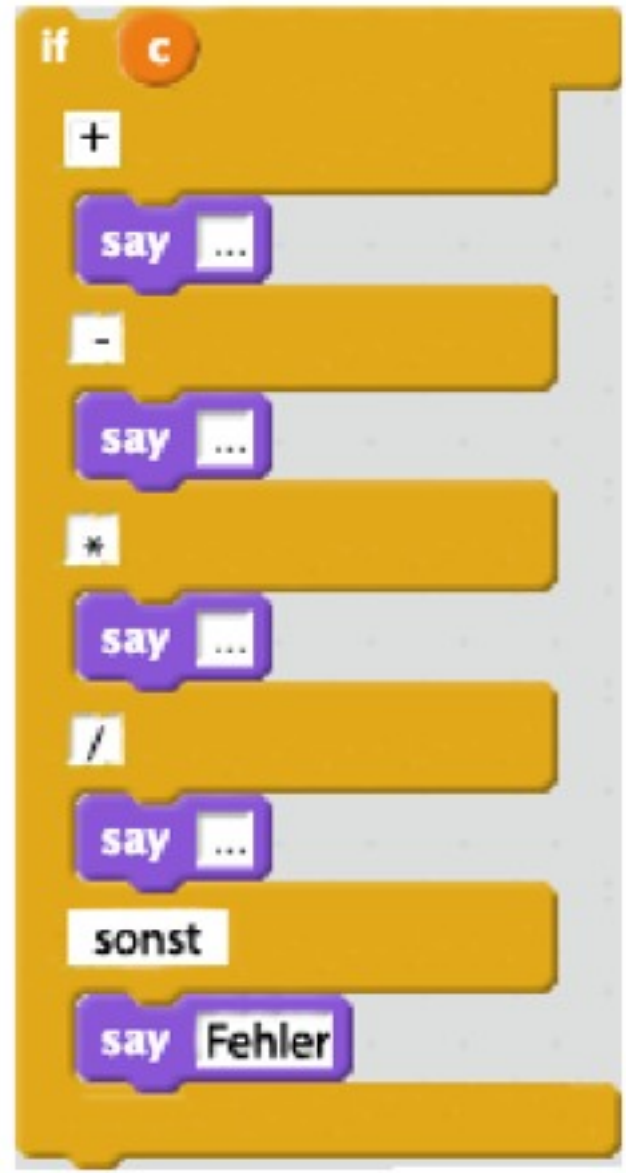




Mehrfachverzeigung

- Allgemein:

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



Mehrfachverzeigung

- Allgemein:

```
switch switch_ausdruck
  case case_ausdruck l
      Anweisungen;
  ...
```

- *switch_ausdruck*:
  Skalar (Variable mit einfachem Wert, d.h. kein Vektor) oder Zeichenkette

- *case_ausdruck*:
  Konstante oder Skalar oder
  Array in der Form { *el1*, *el2*, ... }

Mehrfachverzweigung

```
op1 = input('Erster Operand: ');
c = input('Operation: ', 's');
op2 = input('Zweiter Operand: ');
x = '-';
switch c
    case '+'
      fprintf( '%f',  op1 + op2 );
    case x
      fprintf( '%f',  op1 - op2 );
    case {'*', '.'}
      fprintf( '%f',  op1 * op2 );
    case '/'
      fprintf( '%f',  op1 / op2 );
    otherwise
      fprintf( 'Fehler: Falscher Op.\n');
end
```

Konstante

Skalar

Array

Schleifen: Beispiel ggT berechnen

```
EUCLID_OLD(a,b):
  wenn a = 0
  dann gib b aus
  sonst
      solange b ≠ 0
          wenn a > b
          dann a = a - b
          sonst b = b - a
      gib a aus
```

Übungsaufgabe

Schleifen: Collatz' (3*n*+1) Folge

Lothar Collatz, 1937

Algorithmus:

1. Beginne mit einer natürlichen Zahl, *n*
2. Wenn *n* == 1: STOP
3. Wenn *n* gerade: *n* = *n* / 2
4. Sonst: *n* = 3*n* + 1
5. Fahre mit Schritt 2 fort

Es ist nicht bekannt, ob dieser Algorithmus für jedes *n* anhält!

Aufgabe: Collatz' (3*n*+1) Folge

```
n = input('n: ');
while ( n ~= 1 )
    if mod( n, 2 ) ~= 0
        n = 3*n + 1;
    else
        n = n / 2;
    end
end
fprintf("Ende\n");
```

Matlab Funktion `mod`: berechnet den Divisionsrest (modulo)

Aufgabe

- Alle Zahlen von 1 bis 10 ausgeben
- Lösung mit `while`:

```
n = 1;
while n <= 10
    fprintf('%d\n', n);
    n = n + 1;
end
```

Aufgabe

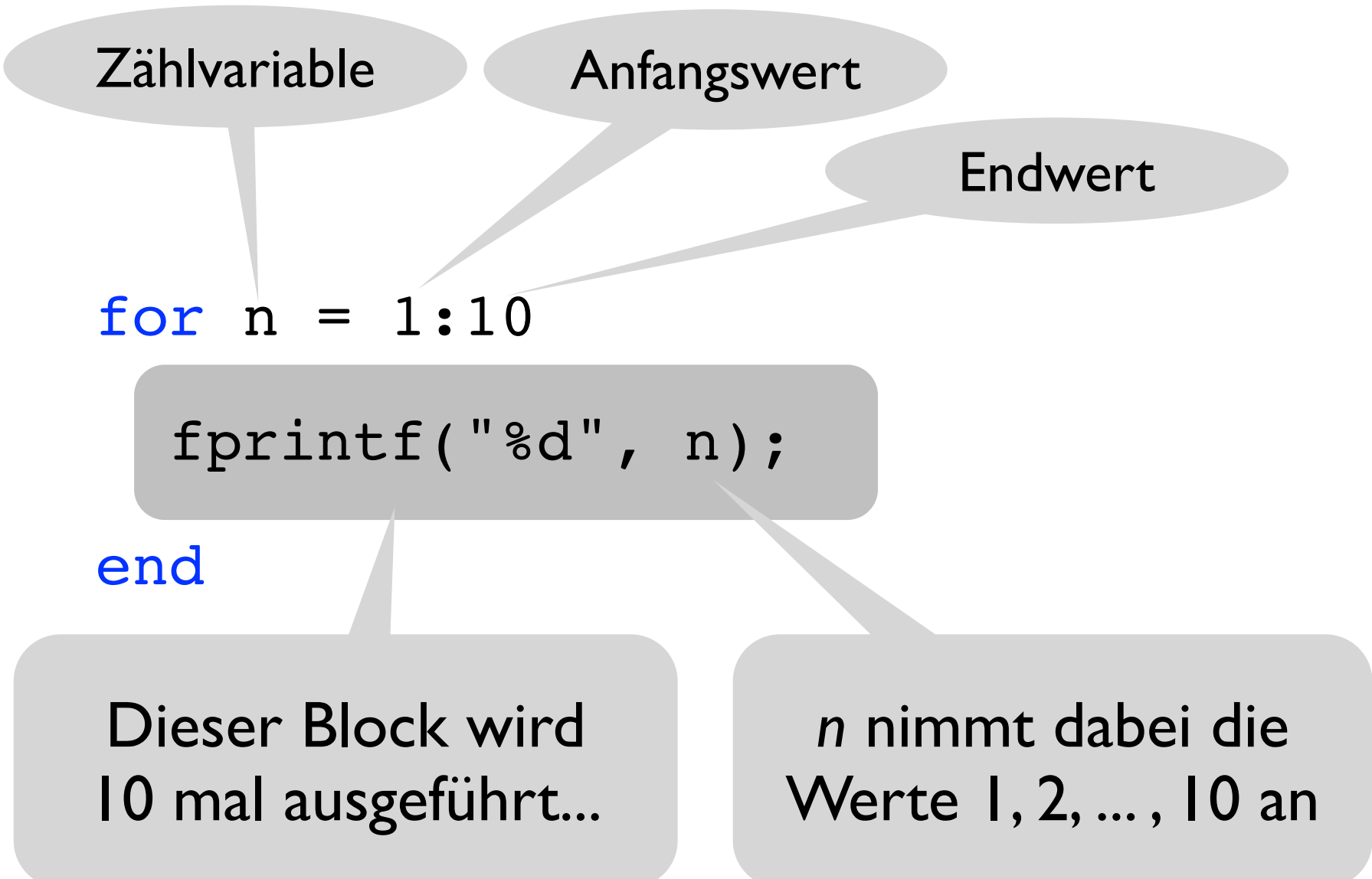
- Alle Zahlen von 1 bis 10 ausgeben
- Lösung mit `for`:

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

Kontrollstrukturen: FOR



Andere Bereichsangaben

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

n nimmt die Werte 1,3,5,7,9 an

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

n nimmt die Werte 10, 9, 8, ... , 0 an

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

n nimmt die Werte 4,8,15,16,23,42 an

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

Manchmal findet man Unsinn...

```
for(j=0;j<50;j++) {
    if(j==0 || j==49)
        spielfeld[i][j] = '#';
    else
        spielfeld[i][j] = ' ';
}
```

Sinnvoller:

```
spielfeld[i][0] = '#';
for(j=1; j<49; j++){
  spielfeld[i][j] = ' ';
}
spielfeld[i][49] = '#';
```

Spezialfälle aus der Schleife herausnehmen!

Beispiel: Fakultät berechnen

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$$

```
n = input('n: ');
fak = 1;
for k = 2:n
    fak = fak * k;
end
fprintf('%d\n', fak);
```

Aufgabe:“Tannenbaum”

```
Wie hoch? 3
*
**
***
```

```
Wie hoch? 4
*
**
***
****
```

```
h = input('Wie hoch? ');
for reihe = 1:h
    for sterne = 1:reihe
        fprintf('*');
    end
    fprintf('\n');
end
```

Aufgabe: Zahlenrätsel

Geocache Description:

Hat nicht jeder schon einmal davon geträumt: Sich die Lottozahlen der nächsten Woche selbst zu berechnen?
Genau das machen wir in diesem Cache. Gesucht sind sechs paarweise verschiedene natürliche Zahlen, die größer als 0 und kleiner als 50 sind. Für diese Zahlen a, b, c, d, e, f gelten folgende Bedingungen:

1. a < b < c < d < e < f
2. a + b + c + d + e + f = 161
3. a * b * c * d * e * f = 138607200

Der Cache befindet sich bei
N 52° 28.(b - a) * b + d - c
E 013° 32.a * (f - e) * (d - c)

Aufgabe: Zahlenrätsel

Gesucht sind sechs natürliche Zahlen *a*,*b*,*c*,*d*,e,*f*.
Für diese Zahlen gelten folgende Bedingungen:

1.) $0 < a < b < c < d < e < f < 50$

2.) $a + b + c + d + e + f = 161$

3.) $a * b * c * d * e * f = 138607200$

Aufgabe: Zahlenrätsel

```
for a = 1:49
  for b = a+1:49
    for c = b+1:49
      for d = c+1:49
        for e = d+1:49
          for f = e+1:49
            if (a+b+c+d+e+f == 161) && (a*b*c*d*e*f == 138607200)
                fprintf('a=%d,b=%d,c=%d,d=%d,e=%d,f=%d\n', ...
                    a, b, c, d, e, f);
            end
          end
        end
      end
    end
  end
end
```

Sprünge in Schleifen

- `continue`
  - Überspringt den Rest des Schleifenblocks, beginnt neuen Durchlauf der Schleife
- `break`
  - Bricht Schleife ab