E Ablaufkontrolle

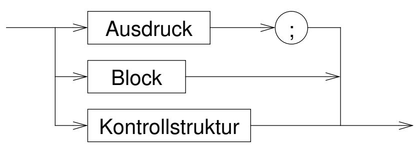
- Struktur eines C-Hauptprogramms
- Anweisungen und Blöcke
- Bedingte Anweisung
  - einfache Verzweigung
  - mehrfache Verzweigung
- Fallunterscheidung
- Schleifen
  - abweisende Schleife
  - nicht abweisende Schleife
  - Laufanweisung
  - Schleifensteuerung

E.1 Struktur eines C-Hauptprogramms

```
main()
{
    Variablendefinitionen
    Anweisungen
}
```

E.2 Anweisungen

Anweisung:



E.3 Blöcke

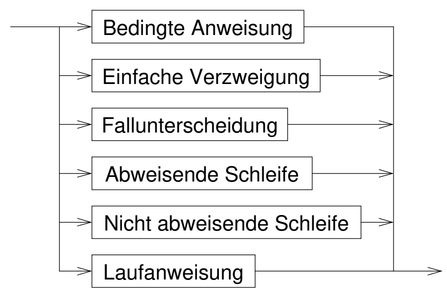
- Zusammenfassung mehrerer Anweisungen
- Lokale Variablendefinitionen → Hilfsvariablen
- Schaffung neuer Sichtbarkeitsbereiche (***Scopes***) für Variablen
  - bei Namensgleichheit ist immer die Variable des innersten Blocks sichtbar

```
main()
{
    int x, y, z;
    x = 1;
    {
        int a, b, c;
        a = x+1;
        {
            int a, x;
            x = 2;
            a = 3;
        }
        /* a: 2, x: 1 */
    }
}
```

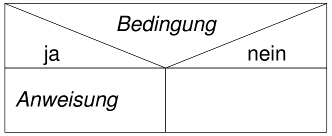

E.4 Kontrollstrukturen

- Kontrolle des Programmablaufs in Abhängigkeit von dem Ergebnis von Ausdrücken

Kontrollstruktur:

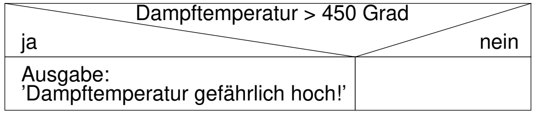


1 Bedingte Anweisung



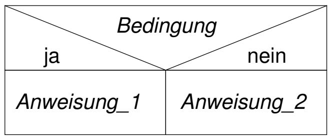

```
if ( Bedingung )
    Anweisung
```

- Beispiel:



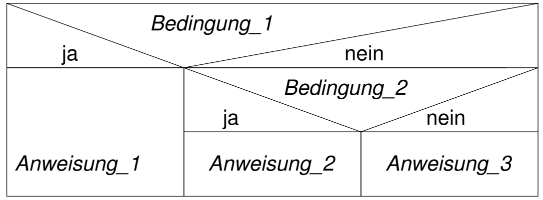

```
if (temp >= 450.0)
  printf("Dampftemperatur gefaehrlich hoch!\n");
```


1 Bedingte Anweisung
einfache Verzweigung




```
if ( Bedingung )
    Anweisung_1
else
    Anweisung_2
```

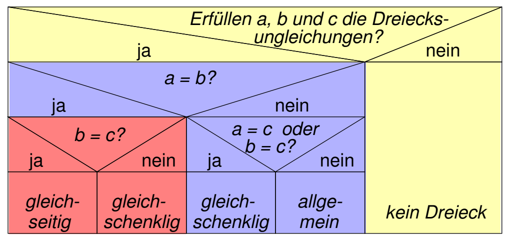

1 Bedingte Anweisung
mehrfache Verzweigung




```
if ( Bedingung )
    Anweisung_1
else if ( Bedingung_2 )
    Anweisung_2
else
    Anweisung_3
```


1 Bedingte Anweisung
mehrfache Verzweigung (2)

- Beispiel: Eigenschaften von Dreiecken — Struktogramm



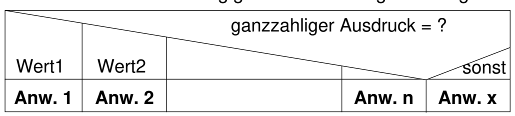
1 Bedingte Anweisung mehrfache Verzweigung (3)

- Beispiel: Eigenschaften von Dreiecken — Programm

```
printf("Die Seitenlaengen %f, %f und %f bilden ", a, b, c);

if ( a < b+c && b < a+c && c < a+b )
    if ( a == b )
        if ( b == c )
            printf("ein gleichseitiges");
        else
            printf("ein gleichschenkliges");
    else
        if ( a==c || b == c )
            printf("ein gleichschenkliges");
        else
            printf("ein allgemeines");
else
    printf("kein");
printf(" Dreieck");
```


2 Fallunterscheidung

- Mehrfachverzweigung = Kaskade von if-Anweisungen
- verschiedene Fälle in Abhängigkeit von einem ganzzahligen Ausdruck




```
switch ( Ausdruck ) {
  case Wert_1:
          Anweisung_1
          break;
  case Wert_2:
          Anweisung_2
          break;
  . . .
  case Wert_n:
          Anweisung_n
          break;
  default:
          Anweisung_x
}
```

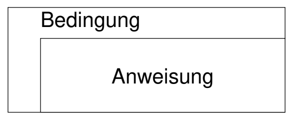

2 Fallunterscheidung — Beispiel

```
#include <stdio.h>

main()
{
  char zeichen;
  int  i;
  int ziffern, leer, sonstige;

  ziffern = leer = sonstige = 0;

  while ((zeichen = getchar()) != EOF)
    switch (zeichen) {
      case '0':
      case '1':
      case '2':
      case '3':
      case '4':
      case '5':
      case '6':
      case '7':
      case '8':
      case '9':          ziffern++;
                         break;
      case ' ':
      case '\n':
      case '\t':         leer++;
                         break;
      default:           sonstige++;
    }

  printf("Zahl der Ziffern = %d\n", ziffern);
  printf("Zahl der Leerzeichen = %d\n", leer);
  printf("Zahl sonstiger Zeichen = %d\n", sonstige);
}
```

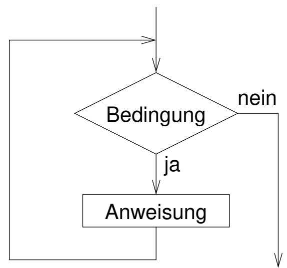

3 Schleifen

- Wiederholte Ausführung von Anweisungen in Abhängigkeit von dem Ergebnis eines Ausdrucks

4 abweisende Schleife

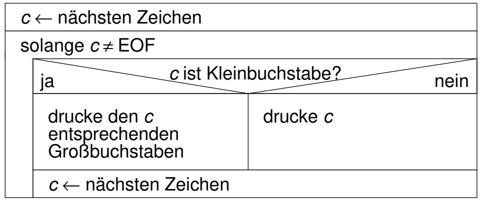



```
while ( Bedingung )
    Anweisung
```



4 abweisende Schleife (2)

- Beispiel: Umwandlung von Klein- in Großbuchstaben

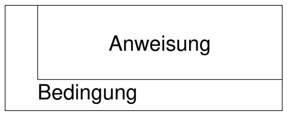



```
char c;
c = getchar();
while ( c != EOF ) {
    if ( c >= 'a' && c <= 'z' )
        putchar(c+'A'-'a');
    else
        putchar(c);
    c = getchar();
}
```

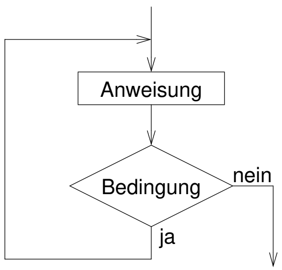
➤ abgekürzte Schreibweise

```
while ( (c = getchar()) != EOF )
    if ( c >= 'a' && c <= 'z' )
        putchar(c+'A'-'a');
    else
        putchar(c);
```


5 nicht-abweisende Schleife




```
do
    Anweisung
while ( Bedingung )
```



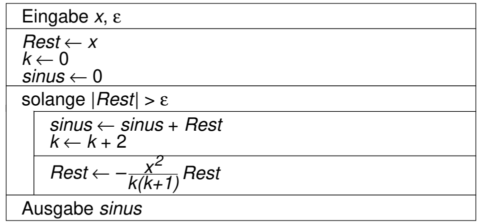
6 Schleifen — Bsp. Sinusberechnung

- Taylor-Reihe: $\sin x = \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{(2i+1)!} + r_{n+1}(x)$

wobei: $|r_{n+1}(x)| \le |R_n(x)|$ mit $R_n(x) = (-1)^n \frac{x^{2n+1}}{(2n+1)!}$

Also: $\sin x = \sum_{i=0}^{n} R_i(x) + r_{n+1}(x)$

mit $R_n(x) = x \cdot \left(-\frac{x^2}{2 \cdot 3}\right) \cdot \left(-\frac{x^2}{4 \cdot 5}\right) \cdots \left(-\frac{x^2}{(2n) \cdot (2n+1)}\right)$



6 Schleifen — Bsp. Sinusberechnung (2)

- Programm

```
#include <stdio.h>
#include <math.h>

main()
{
        double x;                       /* Argument */
        double sinus;                   /* Summenwert */
        double x_quadrat;               /* x*x */
        double rest;                    /* Summenglied */
        double eps;                     /* Genauigkeit */
        int k;                          /* Nenner */

        printf("Berechnung des Sinus von ");
        scanf("%lf", &x);
        printf("Die Genauigkeit soll sein: ");
        scanf("%lf", &eps);

        k = 0;
        sinus = 0.0;
        rest = x;
        x_quadrat = x*x;

        while ( fabs(rest) > eps ) {
                sinus += rest;
                k += 2;
                rest *= -x_quadrat/(k*(k+1));
        }

        printf("sin(%lf) = %lf +- %lf\n", x, sinus, fabs(rest));
}
```

7 Laufanweisung

v ← Startausdruck (Inkrement) Endausdruck
Anweisung

```
for (v = Startausdruck; v <= Endausdruck; v += Inkrement)
    Anweisung
```

allgemein:

```
for (Ausdruck_1; Ausdruck_2; Ausdruck_3)
    Anweisung
```

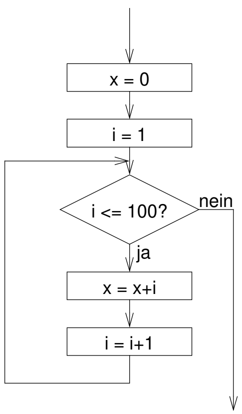
```
Ausdruck_1;
while (Ausdruck_2) {
    Anweisung
    Ausdruck_3;
}
```


7 Laufanweisung (2)

- Beispiel: Berechne $x = \sum_{i=1}^{100} i$

$x \leftarrow 0$
$i \leftarrow 1(1)100$
$x \leftarrow x + i$

```
x = 0;
for ( i=1; i<=100; i++)
    x += i;
```



7 Laufanweisung (3)

- Negativbeispiele
  - $x = \sum_{i=1}^{100} i$

```
for ( x = 0, i = 1; i <= 100; x += i++ );
```

  - Sinusberechnung

```
x_quadrat = x*x;

for ( i = 0, sinus = 0.0, rest = x;
        fabs(rest) > eps;
        i += 2, rest *= -x_quadrat/(i+(i+1) )
    sinus += rest;
```


8 Schleifensteuerung

- break
  - bricht die umgebende Schleife bzw. `switch`-Anweisung ab

```
char c;

do {
    if ( (c = getchar()) == EOF ) break;
    putchar(c);
}
while ( c != '\n' );
```

- continue
  - bricht den aktuellen **Schleifendurchlauf** ab
  - setzt das Programm mit der Ausführung des Schleifenkopfes fort