if und else

- *Verzweigung wird durch eine Bedingung entschieden*

```
if (BEDINGUNG) {
    BLOCK
}else{
    BLOCK
}
```

- „else“-Block ist optional

```
int zahl=9;

if(zahl==7) {
    printf("sieben\n");
}else {
    printf("nicht sieben\n");
}
```

```
nicht sieben
```

if und else – Verschachtelung

- sollen mehrere verschiedene Fälle nacheinander geprüft werden, so kann man die if-Anweisung schachteln

```
int zahl=8;

if(zahl==7) {
    printf("sieben\n");
}else {
    if(zahl==8) {
        printf("acht\n");
    }else {
        printf("nicht sieben und nicht acht\n");
    }
}
```

```
acht
```

if und else – Verkettung

- sollen mehrere verschiedene Fälle nacheinander geprüft werden, so kann man die if-Anweisung verketten

```
int zahl=8;

if(zahl==6) {
    printf("sechs\n");
}else if(zahl==7) {
    printf("sieben\n");
}else if(zahl==8){
    printf("acht\n");
}else{
    printf("nicht sechs, nicht sieben und nicht acht\n");
}
```

```
acht
```

Vergleichsoperatoren

- damit ein if-Block ausgeführt wird, muss die Bedingung zwischen den Klammern wahr sein
  - wahr: Ausdruck nicht 0 (Null)
  - falsch: Ausdruck gleich 0 (Null)

```
if(1) printf("1 ist wahr\n");
if(0) printf("0 ist wahr\n");
if(4711) printf("4711 ist wahr\n");
if(1-1) printf("1-1 ist wahr\n");
```

```
1 ist wahr
4711 ist wahr
```

- Die Bedingung kann man auch durch einen Vergleich formulieren.

Vergleichsoperatoren – ist gleich und ungleich

- Überprüfung ob zwei Werte **gleich** sind durch doppeltes Gleichheitszeichen (==)
  - somit von einer Zuweisung unterscheidbar (=)
- Überprüfung ob zwei Werte **ungleich** sind durch Ausrufezeichen + Gleichheitszeichen (!=)

```
int a=5;
if(a == 5) printf("a ist fuenf\n");
if(a != 5) printf("a ist nicht fuenf\n");
```

```
a ist fuenf
```

Vergleichsoperatoren – größer und größer gleich

```
int a=5;
if(a > 5) printf("a ist groesser fuenf\n");
if(a >= 5) printf("a ist fuenf oder groesser fuenf\n");
```

```
a ist fuenf oder groesser fuenf
```

Vergleichsoperatoren – kleiner und kleiner gleich

```
int a=5;
if(a < 5) printf("a ist kleiner fuenf\n");
if(a <= 5) printf("a ist fuenf oder kleiner fuenf\n");
```

```
a ist fuenf oder kleiner fuenf
```

Logische Operatoren

- Negation durch Ausrufezeichen (!)

```
if(!0) printf("aus falsch wird wahr, 0 -> 1\n");
```

```
aus falsch wird wahr, 0 -> 1
```

- UND-Verknüpfung
  - mit && können mehrere Bedingungen auf Erfüllung überprüft werden

```
int a=0, b=3;
if(!a && b > 1) {
    printf("a ist nicht wahr und b ist groesser 1\n");
}
```

```
a ist nicht wahr und b ist groesser 1
```

Logische Operatoren

- ODER-Verknüpfung
  - wenn nur eine von mehreren Bedingungen erfüllt sein muss, wird die ODER-Verknüpfung verwendet (||)

```
int a=0, b=1;
if(a || b) {
    printf("a oder b ist wahr\n");
}
```

```
a oder b ist wahr
```

Logische Operatoren

- *Exklusive* ODER-Verknüpfung (XOR)
  - nur eine der Bedingungen darf erfüllt sein, die andere Bedingung muss dann falsch sein (^).

```
int a=0, b=1;
if(a ^ b) {
    printf("[1] Die Bedingung ist wahr\n");
}

a=1;
// jetzt sind beide Variablen nicht 0, daher ist die
// XOR-Bedingung nicht mehr wahr
if(a ^ b) {
    printf("[2] Die Bedingung ist wahr\n");
}
```

```
[1] Die Bedingung ist wahr
```

switch case

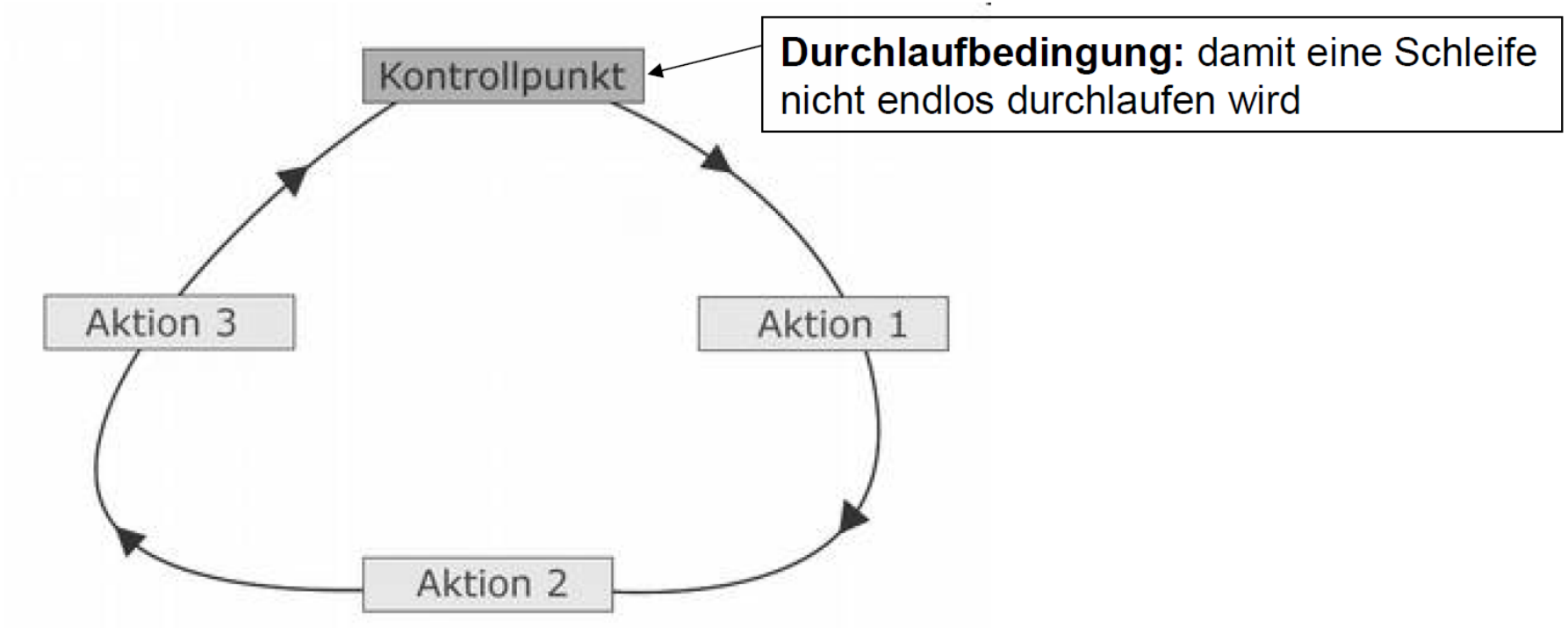
- für die Unterscheidung von vielen Fällen
  - `switch(Ausdruck)`
  - `case X:` für die einzelnen Fälle, nach dem Doppelpunkt folgen die Anweisungen
  - `break` schließt einen case-Block ab
  - `default`-Block: wenn kein Fall erreicht wird, wird der default-Block ausgeführt

```
int a=2;
switch(a) {
    case 1: printf("a ist eins\n");
    break;
    case 2: printf("a ist zwei\n");
    break;
    case 3: printf("a ist drei\n");
    break;
    default: printf("a ist irgendwas\n");
    break;
}
```

```
a ist zwei
```

Schleifen

- werden verwendet, um Wiederholungen im Programm zu realisieren
- auch Wiederholungsstrukturen oder Iterationen genannt
- Programm läuft nicht von oben nach unten, sondern springt zurück und wiederholt einen Programmteil mehrmals



while Schleife

- es sollen die Zahlen von eins bis fünf auf dem Bildschirm ausgegeben werden:

```
printf("Zahl 1\n");
printf("Zahl 2\n");
printf("Zahl 3\n");
printf("Zahl 4\n");
printf("Zahl 5\n");
```

- Anzahl der Programmzeilen entspricht Anzahl der Ausgaben
  - bei der Ausgabe der Zahlen von 1 bis 100 wären das schon 100 Zeilen

```
int i=1;
while(i <= 100) {
    printf("Zahl %d\n", i);
    i++;
}
```

```
Zahl 1
Zahl 2
Zahl 3
...
...
Zahl 98
Zahl 99
Zahl 100
```

Endlosschleife mit while

- einzige Möglichkeit zum Beenden der Schleife: **`break`**

```
#include <stdio.h>

int main(void) {
   int zahl, summe=0;
   printf("Summenberechnung\nBeenden der Eingabe mit 0 \n");

   while(1) {    /* Endlosschleife, denn: 1 ist immer wahr */

      printf("Bitte Wert eingeben > ");
      scanf("%d", &zahl);

      if(zahl == 0)     /*  Haben wir 0 eingegeben ...?   */
         break;         /* ... dann raus aus der Schleife */
      else
         summe+=zahl;
   }
   printf("Die Summe aller Werte beträgt: %d\n", summe);

   return 0;
}
```

for Schleife

- Zähler-gesteuerte Schleife
  - es wird eine Variable benötigt, die die Anzahl der Durchläufe zählt
  - Üblicherweise werden diese Zähl-Variablen beginnend mit dem Buchstaben i benannt (i, j, k etc.)
- Schleife wird mit Schlüsselwort **`for`** eingeleitet
- In der Klammer gibt es drei Bereiche, jeweils getrennt durch ein Semikolon:
  - Bereich 1: Startwert der Zählvariablen setzen, z.B. i=0
  - Bereich 2: Durchlauf-Bedingung, z.B. i<5
  - Bereich 3: Operation auf Zählvariable ausführen, z.B. i++

```
int i;
for(i=0; i<5; i++) {
    printf("Zahl %d\n", i+1);
}
```

```
Zahl 1
Zahl 2
Zahl 3
Zahl 4
Zahl 5
```

for Schleife - Verschachtelung

- Schleifen können beliebig verschachtelt werden
- Beispiel:
  - äußere Schleife: es sollen zehn Zeilen ausgegeben werden
  - innere Schleife: es sollen in jeder Zeile Sternchen ausgegeben werden, wobei die Anzahl der * der jeweiligen Zeilennummer entspricht (z.B. Zeile 2 hat zwei Sternchen)

```
int i, j;

// Schleife fuer die Zeilen
for(i=0; i<10; i++) {
    printf("\nZeile %2d: ", i+1);

    // Schleife fuer die Spalten
    for(j=0; j<=i; j++) {
        printf("*");
    }
}
printf("\n");
```

```
Zeile 1: *
Zeile 2: **
Zeile 3: ***
Zeile 4: ****
Zeile 5: *****
Zeile 6: ******
Zeile 7: *******
Zeile 8: ********
Zeile 9: *********
Zeile 10: **********
```

do while Schleife

- while- und for-Schleifen sind kopfgesteuert
- do while-Schleifen sind fußgesteuert
  - es wird also zunächst der Schleifenblock durchlaufen und danach erst die Bedingung für einen erneuten Durchlauf geprüft
  - das bedeutet, die Schleife wird mindestens einmal durchlaufen!
- Beispiel:
  - Es soll das Alter eingegeben werden
  - Die Schleife wird erst bei Eingabe eines glaubhaften Wertes (6-99) verlassen

```
int alter;
do {
    printf("\nBitte geben sie ihr Alter ein: ");
    scanf("%d", &alter);
} while(alter < 6 || alter > 99);

printf("Danke.\n");
```

mit Semikolon beenden!

```
Bitte geben sie ihr Alter ein: 2
Bitte geben sie ihr Alter ein: 101
Bitte geben sie ihr Alter ein: 200
Bitte geben sie ihr Alter ein: 33
Danke.
```

break

- mit dem Schlüsselwort **break** kann zu jeder Zeit die Schleife verlassen werden
- Beispiel:
  - Benutzer gibt Summanden zwischen 1 und 50 ein, um eine Gesamtsumme von 100 zu erreichen
  - Ist der eingegebene Summand nicht zw. 1 und 50 wird die Schleife sofort verlassen

```
int summand, summe=0;
do {
    printf("\nSumme = %d", summe);
    printf("\nEingabe von Summand zwischen 1 und 50: ");
    scanf("%d", &summand);

    if(summand < 1 || summand > 50) {
        break;
    }

    summe += summand;
}while(summe < 100);

if(summe < 100)
    printf("\nSie haben die Zahl 100 wegen ungueltigen Eingaben nicht erreicht.\n");
else
    printf("\nSie haben den Wert 100 erreicht.\n");
```

continue

- mit dem Schlüsselwort **continue** kann direkt zum Kontrollpunkt gesprungen werden
  - der restliche Code im Schleifen-Block wird dann nicht mehr ausgeführt
- Beispiel:
  - Abwandlung des break-Beispiels → falsche Eingaben werden nun ignoriert

```
int summand, summe=0;
do {
    printf("\nSumme = %d", summe);
    printf("\nEingabe von Summand zwischen 1 und 50: ");
    scanf("%d", &summand);

    if(summand < 1 || summand > 50) {
        printf("Ungueltige Eingabe");
        continue;
    }

    summe += summand;
}while(summe < 100);
```