

Konzepte des prozeduralen Programmierens

Stand September 2022

- Prof. Dr. Oliver S. Lazar / Christian Frank

3 Kontrollstrukturen

Verzweigungen, Schleifen

Kontrollstrukturen definieren die Reihenfolge, in der Aktionen durchgeführt werden.

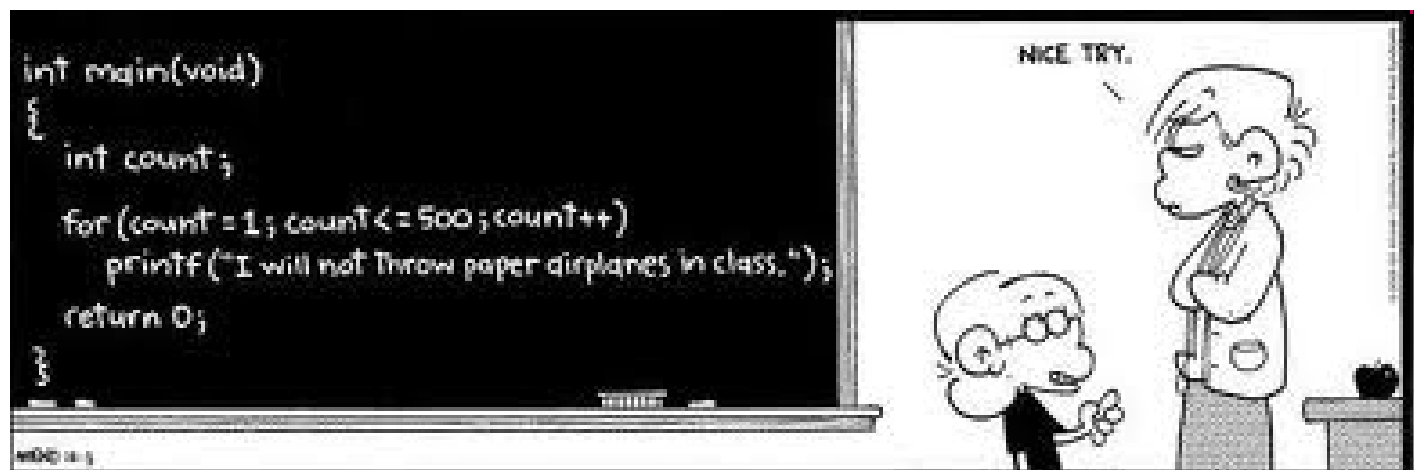
❑ Dieses Kapitel beschreibt

❑ **Verzweigungen**

❑ *if, else, switch/case*

❑ **Schleifen**

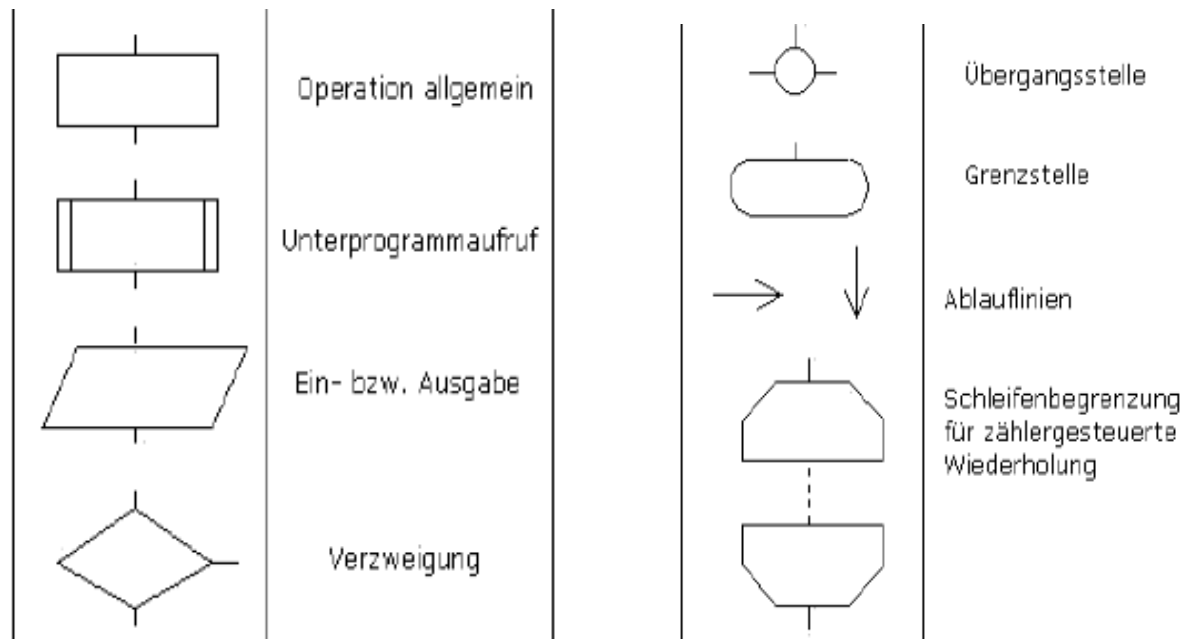
❑ *while, for, do while, break, continue*



Programmablaufplan (PAP)

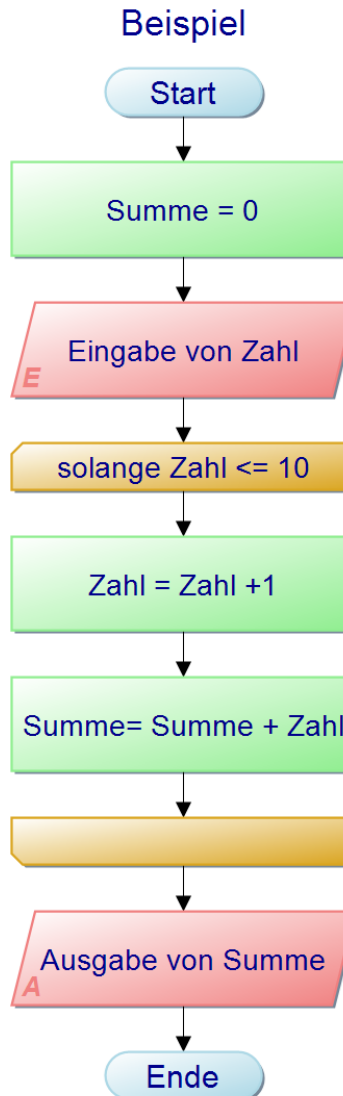
❑ Ablaufdiagramm für ein Computerprogramm

- ❑ auch Flussdiagramm oder Programmstrukturplan
- ❑ graphische Darstellung zur Umsetzung eines Algorithmus
- ❑ beschreibt die Folge von Operationen zur Lösung einer Aufgabe.



Programmablaufplan (PAP)

□ Beispiel



❑ Algorithmus von Euklid

- ❑ zur Bestimmung des größten gemeinsamen Teilers (ggT)

solange $x \neq y$ ist, wiederhole:

wenn $x > y$, dann:

ziehe y von x ab und weise das Ergebnis x zu.

andernfalls:

ziehe x von y ab und weise das Ergebnis y zu.

ENDE solange

x (bzw. y) ist der gesuchte ggT

Was enthält dieser Algorithmus alles?

- Variablen x und y
- Grundoperationen (vergleichen, abziehen, zuweisen)
- **sequentielle Folge** von Anweisungen, aber auch Auswahl (**Selektion**) und Wiederholung (**Iteration**)

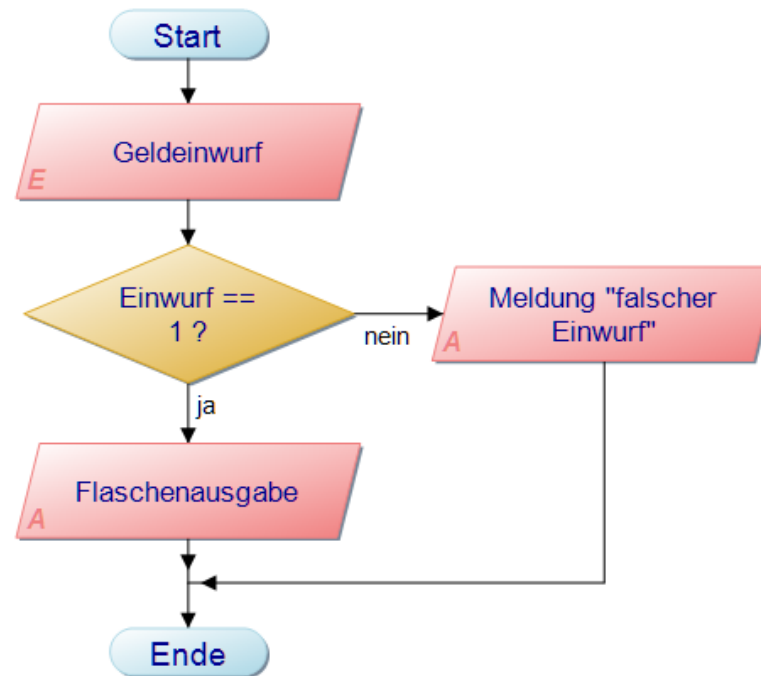
☐ **Aufgabe 03.01**

- ☐ Erstelle den Programmablaufplan für den Algorithmus von Euklid
- ☐ Benutze dafür am besten ein Blatt kariertes Papier

Das Programm soll in Abhängigkeit von der Eingabe reagieren.

Beispiel: Getränke-Automat (zunächst ganz einfach)

- ❑ *Der Automat besitzt nur eine Sorte von Getränken.*
- ❑ *Ein Getränk kostet 1 Euro.*
- ❑ *Ein Getränk kann nur mit einem 1-Euro-Geldstück bezahlt werden.*



Beispiel: Getränke-Automat (zunächst ganz einfach)

- ☐ *Der Automat besitzt nur eine Sorte von Getränken.*
- ☐ *Ein Getränk kostet 1 Euro.*
- ☐ *Ein Getränk kann nur mit einem 1-Euro-Geldstück bezahlt werden.*

```
// Getraenke Automat Version 0.1

int einwurf=0;
printf("Getraenke Automat | Bitte werfen Sie 1 Euro ein: ");
scanf("%d", &einwurf);

// überprüfe Geldstueck
if(einwurf == 1) {
    printf("\nVielen Dank, bitte entnehmen Sie ihr Getraenk.\n");
}else{
    printf("\nSie haben kein 1 Euro Stueck eingeworfen.\n");
}
```

```
Getraenke Automat | Bitte werfen Sie 1 Euro ein: 1

Vielen Dank, bitte entnehmen Sie ihr Getraenk.
```

if und else

- ❑ *Verzweigung wird durch eine Bedingung entschieden*

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

- ❑ „else“-Block ist optional

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

```
sieben
```

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 verketteten

```
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
```

Bitweiser Operator

❑ *Exklusive ODER-Verknüpfung (XOR)*

- ❑ nur eine der Bedingungen darf erfüllt sein, die andere Bedingung muss dann falsch sein (^).
- ❑ **ACHTUNG!** Kein logischer Operator, sondern Bitweiser Operator!

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

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

Bedingungsoperator

- ❑ Kurzschreibweise der **if else** Anweisung
 - ❑ z.B. wenn die Zuweisung eines Wertes von einem Vergleich abhängig ist

Variante mit **if else**:

```
int a=3, b=5, max;  
  
if(a > b) {  
    max = a;  
}else {  
    max = b;  
}  
printf("Maximum: %d\n", max);
```

Variante mit
Bedingungsoperator

```
int a=3, b=5, max;  
max = (a > b)?a:b;  
printf("Maximum: %d\n", max);
```

```
Maximum: 5
```

Aufg. 03.01a

Erstelle das Programm für den Getränkeautomaten und committe es in Git.

Initialisiere Git in der Subdirectory der Aufgabe:

```
$ git init
```

Füge die beiden Dateien hinzu, nachdem Du sie erstellt hast:

```
$ git add *.c  
$ git add Makefile
```

Anschließend committe Deine Dateien:

```
$ git commit -m 'Initial project version'
```

Aufg. 03.02

Lies über die Konsole zwei Zahlen ein und gib auf dem Bildschirm aus, ob die erste Zahl kleiner, größer oder gleich der zweiten Zahl ist.

Aufg. 03.03

Überprüfe, ob in der folgenden if-Konstruktion die printf-Anweisung ausgeführt wird:

```
if( wert1 < wert2 < wert3 ) printf("%d",wert2);
```

Verwende folgende Werte für Ihre Variablen

a) wert1=7, wert2=5, wert3=10

Aufg. 03.04

Überprüfe die Werte für die Wahrheitswerte „wahr“ und „falsch“ mit folgender Anweisung:

```
printf("Wert für wahr: %d\tWert für falsch: %d\n", 2 < 3, 7 != 7);
```

Aufg. 03.05

Lies das Alter eines Menschen ein. Gib dann in Abhängigkeit vom Alter die entsprechende Zeichenkette aus: Kind/Jugendlicher (0 bis unter 18), Erwachsener (18 bis unter 67), Rentner (ab 67).

Aufg. 03.06 (→ später: switch-Aufgabe 03.09)

Nach der Eingabe einer Punktzahl (max. 10) soll die Note als Text ausgegeben werden: 10 – sehr gut, 9 – gut, 8 – befriedigend, 7 – ausreichend, 6 – mangelhaft, <6 – ungenügend

Aufg. 03.07

Lass den Benutzer drei unterschiedliche Zahlen eingeben. Gib auf dem Bildschirm die Zahl aus, die am kleinsten ist.

Aufg. 03.08

Lass den Benutzer drei unterschiedliche Zahlen eingeben. Gib auf dem Bildschirm die Zahl aus, die am größten ist.

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

Aufg. 03.09 → if 03.06

Nach der Eingabe einer Punktzahl (max. 10) soll die Note als Text ausgegeben werden: 10 – sehr gut, 9 – gut, 8 – befriedigend, 7 – ausreichend, 6 – mangelhaft, <6 – ungenügend

Aufg. 03.10

Der Benutzer soll eine Zahl als Kennzeichen für einen Wochentag angeben (1=Montag, 2=Dienstag etc.). Auf dem Bildschirm soll daraufhin der Name des Wochentags erscheinen.

Aufg. 03.11

Abwandlung 03.10: Gib für Montag bis Mittwoch „erste Wochenhälfte“, für Donnerstag und Freitag „zweite Wochenhälfte“ und für Samstag und Sonntag „Wochenende“ aus.

Erweitere den bisherigen Getränke-Automat um folgende Funktionalitäten:

- ☐ Es gibt mehrere Getränke zur Auswahl: Wasser, Limonade und Bier.
- ☐ Wasser kostet 0,50 Euro, Limonade 1 Euro und Bier 2 Euro.
- ☐ Die Getränke werden jeweils mit nur einem Geldstück bezahlt.
- ☐ Verwende ein if/else(if)- und switch-Konstrukt, um den zu zahlenden Betrag und die Zahlung zu überprüfen.

```
Getraenke Automat v0.2
```

```
Waehlen Sie ihr Getraenk aus:
```

```
1) Wasser (0,50 Euro)
```

```
2) Limonade (1,00 Euro)
```

```
3) Bier (2,00 Euro)
```

```
Geben Sie 1, 2 oder 3 ein: 3
```

```
Bitte werfen Sie 2.00 Euro ein: 2.0
```

```
Vielen Dank, bitte entnehmen Sie ihr Getraenk.
```

**Bildschirmausgabe einer
möglichen Lösung**

Verwende Git um die Änderungen zu speichern:

- ☐ Gehe in die Subdirectory von Aufgabe 3.01a
- ☐ Kopiere Dir ein [.gitignore](#)-File für C
- ☐ Ändere den Quelltext ab und committe jeden Schritt

```
$ git commit -m 'First change'
```

Sieh Dir den Status an:

```
$ git status
```

Gibt es Differenzen?

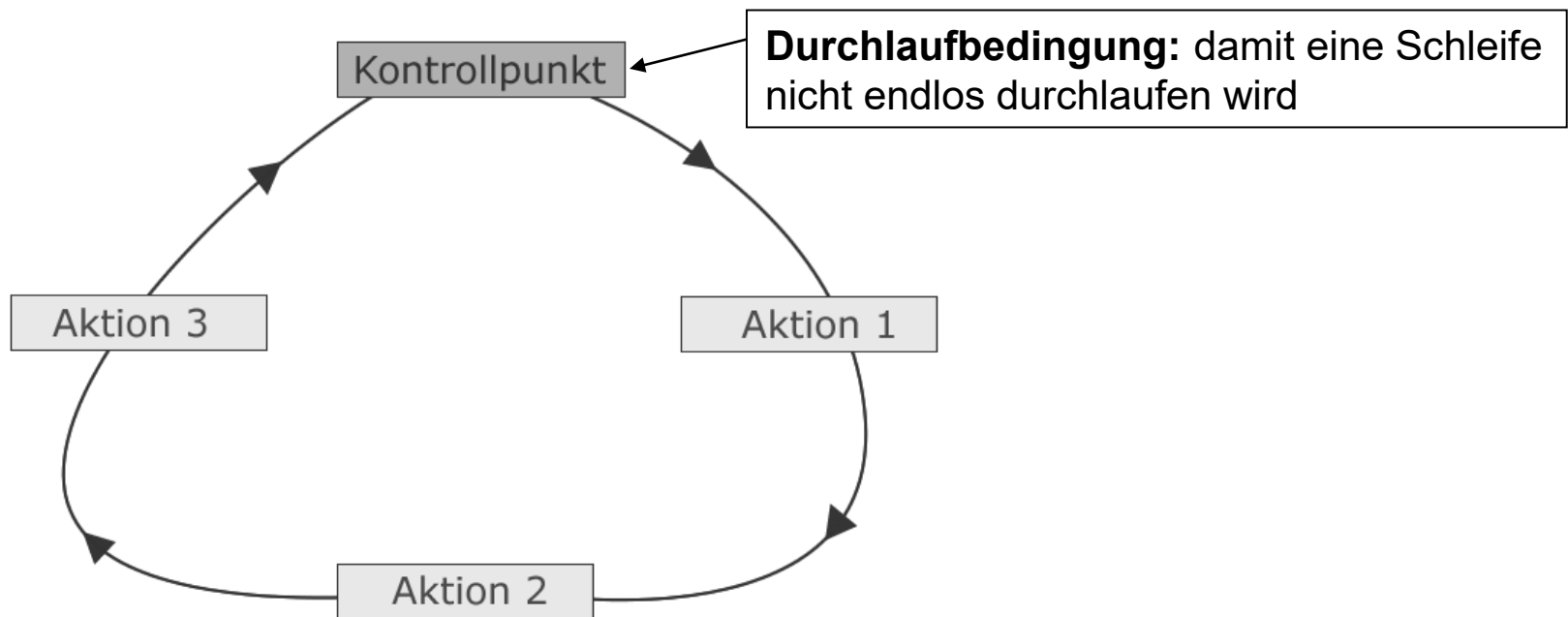
```
$ git diff
```

Historie:

```
$ git log
```

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);
```

... am Beispiel einer for-Schleife

K&R-Stil ▼▲

```
for(i=0; i < 10; i++){  
    /* Anweisungen */  
}
```

Whitesmith-Stil ▼▲

```
for(i=0; i < 10; i++)  
    {  
        /* Anweisungen */  
    }
```

Allman-Stil ▼▲

```
for(i=0; i < 10; i++)  
{  
    /* Anweisungen */  
}
```

Aufg. 03.13

Gib die Zahlen von 1 bis 25 auf dem Bildschirm aus.

Aufg. 03.14

Lass ein Rechteck von 10*10 Zeichen mit Sternchen (*) ausfüllen, verwende dabei zwei verschachtelte Schleifen.

Aufg. 03.15

Lies sukzessive Zahlen von der Konsole ein, bis der Benutzer die Zahl 0 eingibt. Gib dann die größte und die kleinste eingegebene Zahl aus.

Aufg. 03.16

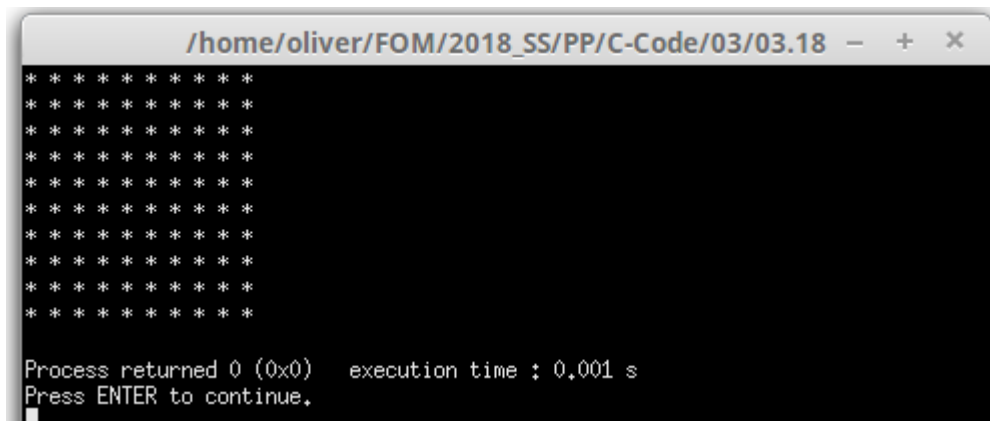
Wie oft kann eine einzugebende Zahl durch fünf ohne Rest geteilt werden?
z.B. 25 → zweimal, 125 → dreimal

Aufg. 03.17

Gib die Zahlen von 1 bis 25 auf dem Bildschirm aus.

Aufg.03.18

Lass ein Rechteck von 10*10 Zeichen mit Sternchen (*) ausfüllen, verwende dabei zwei verschachtelte Schleifen.



```
/home/oliver/FOM/2018_SS/PP/C-Code/03/03.18 - + x
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

Process returned 0 (0x0)   execution time : 0.001 s
Press ENTER to continue.
```

Aufg. 03.19

Gib die Zahlen von 1 bis 25 auf dem Bildschirm aus.

Aufg. 03.20

Lass ein Rechteck von 10*10 Zeichen mit Sternchen (*) ausfüllen.

Aufg. 03.21

Gib das kleine 1*1 auf dem Bildschirm aus.

Aufg. 03.22

Schreibe ein Programm, das die ganzen Zahlen von 1 bis n miteinander multipliziert. Das Programm erwartet die Eingabe von n und berechnet das Produkt von $1*2*3*\dots*n$. In der Mathematik heißt dieses Produkt n Fakultät, geschrieben $n!$.
 $4!$ Entspricht also $1*2*3*4 = 24$

Aufg. 03.23

Schreibe ein Programm, das die Fibonacci-Zahl f_n ausgibt. Die Fibonacci-Folge ist eine unendliche Folge von Zahlen, bei der sich die jeweils folgende Zahl durch Addition ihrer beiden vorherigen Zahlen ergibt. Errechnet werden können sie mittels ... $1+2=3$, $2+3=5$, $3+5=8$, $5+8=13$. Die Formel lautet also: $F(n) = F(n-1) + F(n-2)$

Es gilt für die ersten beiden Zahlen:

$f_0 = 0$ und $f_1 = 1$

Folge: 0, 1, 1, 2, 3, 5, 8, 13...

Beispiel: $f_6 = f_5 + f_4 = 5 + 3 = 8$

Fibonacci-Zahlen

Ursprung: Beispiel zur mathematischen Populationsdynamik von Kaninchen
Aufgabe von Leonardo von Pisa (= Fibonacci), ca. 1180 – ca. 1250

Die Rekursion für die Folge $F(0)$, $F(1)$, ...

Def.: $F(0) = 0$

$$F(1) = 1$$

$$F(n) = F(n-1) + F(n-2) \quad , \quad n \geq 2$$

... generiert die Folge

| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|-------------|---|---|---|---|---|---|---|----|----|----|----|----|-----|
| F(n) | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 |

Erweitere den bisherigen Getränke-Automat um folgende Funktionalitäten:

- ☐ Nach der Wahl der Sorte kann der Benutzer auch eine Menge eingeben. Der zu zahlende Betrag errechnet sich dann aus dem Produkt von Menge und Preis pro Flasche. Die Ausgabe der einzelnen Flaschen erfolgt mit einer Bildschirmausgabe, siehe unten.
- ☐ Die Bezahlung soll jetzt auch mit verschiedenen Geldstücken gemacht werden können. Der Benutzer wird solange aufgefordert Geld einzuwerfen, bis der zu zahlende Betrag erreicht ist.
- ☐ Wähle für beide neuen Funktionen passende Schleifen.

Getraenke Automat v0.3
Waehlen Sie ihr Getraenk aus:
1) Wasser (0,50 Euro)
2) Limonade (1,00 Euro)
3) Bier (2,00 Euro)
Geben sie 1, 2 oder 3 ein: 3
Geben sie die gewuenschte Menge ein: 2

--- Bezahlvorgang ---
Es fehlen noch 4.00 Euro.
Bitte werfen Sie ein Geldstueck ein: 2
Es fehlen noch 2.00 Euro.
Bitte werfen Sie ein Geldstueck ein: 1
Es fehlen noch 1.00 Euro.
Bitte werfen Sie ein Geldstueck ein: 0.5
Es fehlen noch 0.50 Euro.
Bitte werfen Sie ein Geldstueck ein: 0.5

--- Getraenkeausgabe ---
Flasche 1 von 2 wurde ausgegeben.
Flasche 2 von 2 wurde ausgegeben.

Vielen Dank, bitte entnehmen Sie ihre Getraenke.

**Bildschirmausgabe
einer möglichen Lösung**