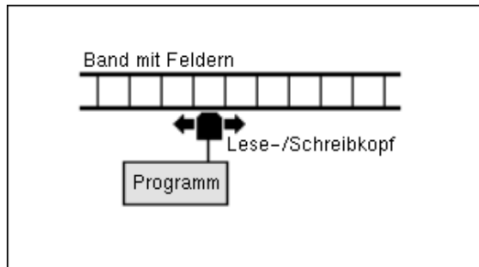


# 3. Übung – Schleifen

- 1 Ideen für Hausaufgabe 1
- 2 Wiederholung Variable und Blöcke
- 3 Schleifen und Schuhgröße
- 4 Schleifen
- 5 Aufgaben
  - Summe und Mittelwert
  - Umrechnung Grad- in Bogenmaß
  - Berechnung der Ziffern im Oktalsystem

# 1. Ideen für Hausaufgabenblatt 1

Turing-Maschine: (Konzept in der theoretischen Informatik)



$$M = (\Sigma, A, \delta, \sigma_0, F)$$

$\Sigma$ : Zustände

$A$ : Alphabet (Zeichen auf dem Band)

$\delta : \Sigma \times A \rightarrow \Sigma \times A \times \{L, R, 0\}$ :

Übergangsfunktion

$\sigma_0 \in \Sigma$ : Anfangszustand

$F \subseteq \Sigma$ : Endzustände

Beispiel: Gesucht Turingmaschine, die die Zeichenkette „111=“ in „0111.“ transformiert. Festlegen des 5-Tupels!

# 1. Ideen für Hausaufgabenblatt 1

$\Sigma = \{s_1, s_2, s_3, s_4\}$  ( Am Anfang unklar wieviel wir benötigen)

$A = \{„0“, „1“, „=“, „“, „.“\}$  ( Alle Zeichen, die bei Ein-und Ausgaben vorkommen)

$\sigma_0 = s_1$  ( Startzustand)

$F = \{s_4\}$  ( Ein Endzustand  $s_4$ )

Übergangsfunktion:  $\sigma: \Sigma \times A \rightarrow \Sigma \times A \times \{L,R,O\}$

( Für ein im Zustand gelesenes Zeichen, wird eine neuer Zustand, das zu schreibende Zeichen und anschließend die Bewegungsrichtung des Leseschreibkopfes festgelegt )

Zustand	Eingabe- zeichen	Neuer Zustand	Ausgabe- zeichen	Kopf- bewegung
$s_1$	1	$s_2$	0	R
$s_2$	1	$s_2$	1	R
$s_2$	=	$s_3$	1	R
$s_3$	(Leerzeichen)	$s_4$	.	O

Definition einer Turingmaschine als ein Beispiel

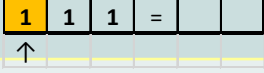
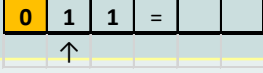
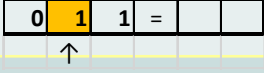
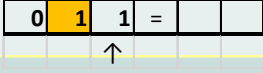
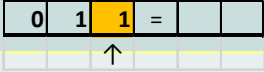
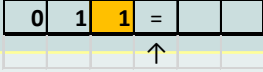
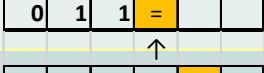
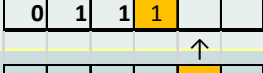
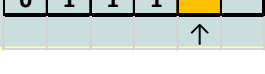
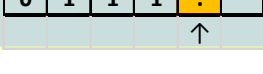
Imperative Programmierung - Variablen und Schleifen

# 1. Ideen für Hausaufgabenblatt 1

Testen für  $111=$  ( Was ist Ergebnis und erreichen wir Endzustand?)

Zustand	Eingabezeichen	Neuer Zustand	Ausgabezeichen	Kopfbewegung
$s_1$	1	$s_2$	0	R
$s_2$	1	$s_2$	1	R
$s_2$	=	$s_3$	1	R
$s_3$	(Leerzeichen)	$s_4$	.	O

Protokoll der Schritte, Zustände und Bandsituation:

	Eingabe		Ergebnis			Vorher	Nachher
1.	$s_1$	1	$s_2$	0	R		
2.	$s_2$	1	$s_2$	1	R		
3.	$s_2$	1	$s_2$	1	R		
4.	$s_2$	=	$s_3$	1	R		
5.	$s_3$		$s_4$	.	O		

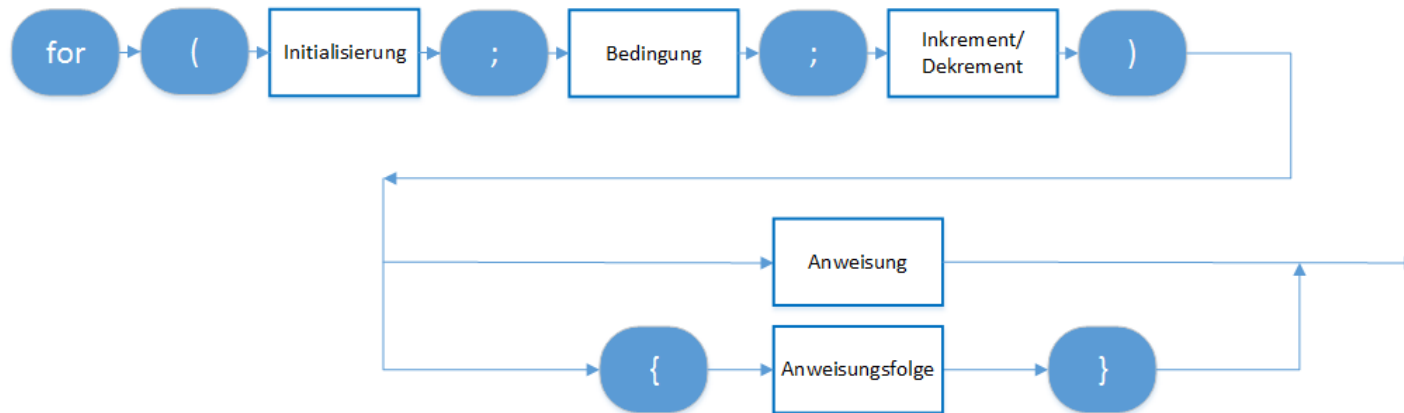
Ergebnis: 0111. und wir erreichen Endzustand  $s_4$

# 1. Syntaxdiagramme

Beispiel for-Schleife in Programmiersprache C als Syntaxdiagramm

**(Eine Anweisungsfolge mehrfach (n-mal) durchlaufen)**

Terminale Elemente (Blau müssen) vorkommen, Nichtterminale können/müssen noch erklärt werden.



**Aufgabe: Zahlen von 1 bis 100 Aufsummieren**

**Lösung in C:**

```
int i;           /* Variable i deklarieren */
int sum= 0;      /* Variable sum deklarieren und initialisieren mit 0 */
for(i=1;i<=100; i++) /* Schleife über i, die 100 mal abgearbeitet wird */
    sum=sum+i;    /* Bisherige Summe um den Wert i erhöhen und speichern */
```

# 1. EBNF

Beispiel Erweiterte Backus-Naur-Form (EBNF) für eine Buch.

(Eine Bücherliste besteht aus mehreren Büchern. Ein Buch hat einen Titel, kann einen oder mehrere Autoren oder einen Herausgeber haben. Es ist der Erscheinungsort, gegebenenfalls ein Verlagsname und das Erscheinungsjahr anzugeben.)

"`ttt`" : terminales Symbol, ist in dieser Form Bestandteil des Satzes einer Sprache.

[ ] : optionales (null- oder einmaliges) Auftreten der geklammerten Elemente

{ } : null- oder beliebig oftmaliges Auftreten der geklammerten Elemente

( ) : Gruppierung von Elementen (einmaliges Auftreten)

| : Trennung von Alternativen

= : Trennung linker von rechter Regelseite.

. : Ende der Regel

*Bezeichner* : Nichtterminal, wird durch weitere Regeln erklärt.

# 1. EBNF

Beispiel Erweiterte Backus-Naur-Form (EBNF) für eine Buch.

(Eine Bücherliste besteht aus mehreren Büchern. Ein Buch hat einen Titel, kann einen oder mehrere Autoren oder einen Herausgeber haben. Es ist der Verlagsort, gegebenenfalls ein Verlagsname und das Erscheinungsjahr anzugeben.)

" <code>ttt</code> "	: terminales Symbol, ist in dieser Form Bestandteil des Satzes einer Sprache.
[ ]	: optionales (null- oder einmaliges) Auftreten der geklammerten Elemente
{ }	: null- oder beliebig oftmaliges Auftreten der geklammerten Elemente
( )	: Gruppierung von Elementen (einmaliges Auftreten)
	: Trennung von Alternativen
=	: Trennung linker von rechter Regelseite.
.	: Ende der Regel

*Bezeichner* : Nichtterminal, wird durch weitere Regeln erklärt.

**Bücherliste = {Buch}**

**Buch = Titel ({Autor}|Herausgeber) Verlagsort [Verlagsname] Erscheinungsjahr**

**Titel = Zeichenkette**

**Zeichenkette = {Buchstabe|Ziffer|Sonderzeichen}**

...

# 1. Reihenentwicklung Sinus

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

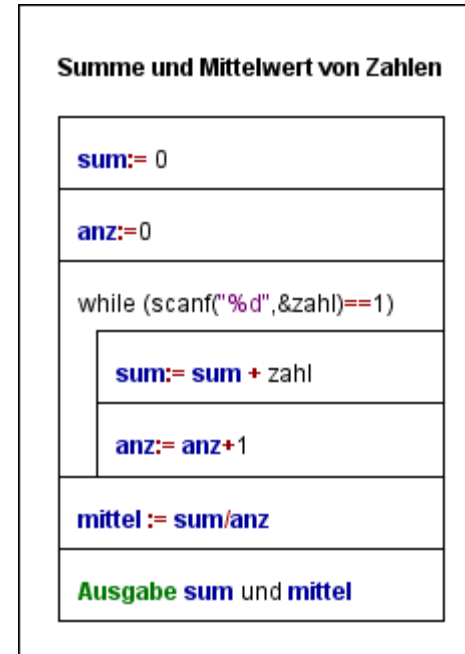
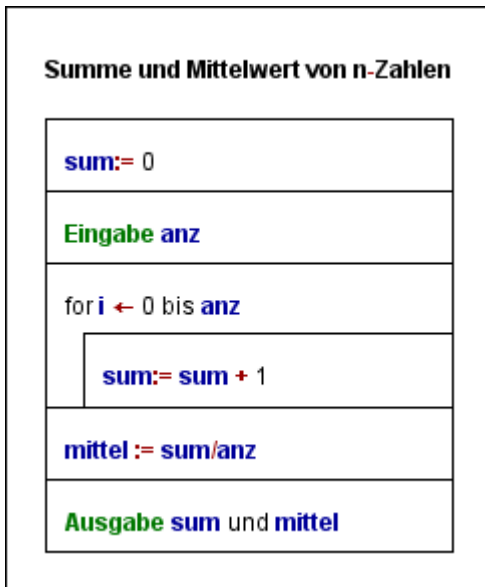
$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$



# 1. Struktogramme

Beispiele für Struktogramme:

- Summation von n-Zahlen und Bestimmung des Mittelwertes bei Vorgabe von n
- Summation von Zahlen bis keine Zahl eingegeben wird und Berechnung Summe und Mittelwert.



## 2. Wiederholung

Verzeichnis (Imp2019) unter dem Laufwerksbuchstaben R: nutzen.

Wir haben kleine C- Programme geschrieben, die Lückentexte enthielten und Nutzung von Variablen enthielten :

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

```
main(){  
    int a=9;  
    double e=16.25;  
    printf(" Die Zahlen %d und %16.3f\n",9, 16.25);  
    printf(" Die Zahlen %d und %16.3f\n", a, e);  
}
```

### Übersetzen und testen von Programmen

```
cd Imp2019
```

```
gcc -g anfang.c -o anfang.exe
```

```
anfang.exe
```

```
< --- wechsele ins Verzeichnis Imp2019  
< --- führe dort C-Compiler aus  
< --- starte das Programm
```

## 2. Wiederholung Variable

Um mit Variablen in der Programmiersprache C arbeiten zu können, muss eine Variable deklariert, definiert und kann initialisiert werden.

- **dekliert** - Name und Datentyp festlegen
- **definiert** - Speicherplatz reservieren
- **initialisiert** - Anfangswert zu weisen

```
int a; // Variable a deklariert, definiert und nicht initialisiert
a = 3; // bestehende Variable a bekommt Wert 3 ( hier Initialisierung)
int b = 9; // Variable b deklariert, definiert und initialisiert mit 9
extern int c; // Variable c nur deklariert, hier keine Speicherung
```

### Ganzzahl Datentypen:

char	(1 Byte-8-Bit)	von	-128 ... +127
short	(2 Byte-16-Bit)	von	-32768 ... +32767
int	(4 Byte-32-Bit)	von	-2147483648 ... +231 - 1
long	(4 Byte-32-Bit)	von	-2147483648 ... +231 - 1

### Gleitkomma Datentypen:

float	(4 Byte-32-Bit)	1.17E-38 .. 3.4E38 (7-8 Stellen)
double	(8 Byte-64-Bit)	2.2E-308 .. 1.7E308(15-16 Stellen)

## 2. Variablen und deren Gültigkeit

Mit Ausnahme des globalen Blocks in einer C-Quelldatei wird eine Block durch “{” “}” eingeschlossen. In Blöcken können eigene Gültigkeitsbereiche für Variablen definiert werden. Nach Beendigung des Blockes kann die dort definiert Variable nicht mehr benutzt werden. Man darf Variablen aus offenen Blöcken nutzen. Blöcke können Namen tragen.

```
#include <stdio.h>
int g, a = 4;
double d=2.1;
qux(){
    int b = 12;
    printf("a=%d, b=%d, g=%d, %f\n",a , b, g, d);/*4, 12, 0 (unbekannt),2.1*/
}
main()
{
    int a = 41,c;
    float b;
    {
        int a= 24;
        b= 7;
        g= a;
        printf("a=%d, b=%d, g=%d, %f\n",a ,(int)b,g,d);/*24, 7, 24 ,2.1 */
    }
    qux();
    printf("a=%d, b=%d, g=%d, %f\n",a ,(int)b,g,d);/*41, 7, 24 ,2.1 */
}
```

# 3. Aufgabe - Berechnung von Schuhgrößen

*Schuhgröße (EU) = (Fußlänge in cm + 1,5) × 1,5*

*Schuhgröße (in Deutschland gebräuchlich) = (Fußlänge in cm + 1,54) / 0.667*

*Brannock-System:*

*Herrengroße (US) = Fußlänge in cm ÷ 2,54 × 3 - 22*

*Damengroße (US) = Fußlänge in cm ÷ 2,54 × 3 - 21*

In der EU sind Schuhgrößen ganzzahlig, die Schuhgrößen im Brannock-System haben Abstufungen von 0.5.

```
#include <stdio.h>
int fusslaenge;
schuhgroesse_de() { /* Hier Deutschland */
}
schuhgroesse_eu() { /* Hier EU */
}
schuhgroesse_usa() { /* Hier Brannock */
}

int main() { /* Eingaben und Ausgaben */
    printf("Eingabe der Fusslaenge in cm\n");
    scanf("%d",&fusslaenge);
    schuhgroesse_de();
    schuhgroesse_eu();
    schuhgroesse_usa();
}
```

# 3. Lösung Schuhgröße

```
#include <stdio.h>

int fusslaenge;

schuhgroesse_de() {
    float schuhgroesse = (int)((fusslaenge + 1.54)/0.667);
    printf("Die Schuhgroesse DE %.0f, bei Fusslaenge von %d cm\n", schuhgroesse, fusslaenge);
}

schuhgroesse_eu() {
    float schuhgroesse = (int)((fusslaenge + 1.5)*1.5);
    printf("Die Schuhgroesse EU %.0f, bei Fusslaenge von %d cm\n", schuhgroesse, fusslaenge);
}

schuhgroesse_usa() {
    float schuhgroesse = (0.5)*(int) (2.0 *((fusslaenge + 1.54)*3/2.54 - 24)); // Herren -24, Damen -23
    printf("Die Schuhgroesse USA %.1f, bei Fusslaenge von %d cm\n", schuhgroesse, fusslaenge);
}

int main() { /* Eingaben und Ausgaben siehe Folie 9 Vorlesung */
    printf("Eingabe der Fusslaenge in cm\n");
    scanf("%d",&fusslaenge);
    schuhgroesse_de();
    schuhgroesse_eu();
    schuhgroesse_usa();
}
```

# 3. Lösung Schuhgröße

## Schritte bei Runden auf 0.5

```
schuhgroesse_usa_Runden_einzeln() {  
    float schuhgroesse = (fusslaenge + 1.54)*3/2.54 - 24; /* Normale Formel*/  
    schuhgroesse = 2 * schuhgroesse; /* Verdoppeln der Schuhgroesse, Werte ueber .5 erzeugen anderen Wert */  
    schuhgroesse = (int) schuhgroesse; /* Schneide Nachkommastellen ab durch Umwandlung in int */  
    schuhgroesse = schuhgroesse/ 2; /*Dividiere durch 2 oder multipliziere mit 0.5, um ggf korrekt auf 0.5 zu kommen */  
    printf("Schuhgroesse USA %.1f (mit runden), bei Fusslaenge von %d cm\n", schuhgroesse, fusslaenge);  
  
    /* Beispiel 25 cm fuehrt zu 7.3 ohne runden,  $7.3 * 2 = 14.6$ , als int-Wert 14 und nach Division in double  
       damit zu 7.0  
       Beispiel 27 cm fuehrt zu 9.7 ohne runden,  $9.7 * 2 = 19.4$ , als int-Wert 19 und nach Division in  
       double damit zu 9.5 */  
}
```

# 4. Schuhgrößen im Bereich

Es sollen die Schuhgrößen für den Bereich von 24 cm bis 38 cm in 1cm Schritten bestimmt werden.

Lösung: Scheife von 24 bis 38 mit 1 als Schrittweite.



# 4. Schuhgrößen im Bereich

Es sollen die Schuhgrößen für den Bereich von 24 cm bis 38 cm in 1cm Schritten bestimmt werden. Lösung: Scheife von 24 bis 38 mit 1 als Schrittweite.

```
int main() {
    fusslaenge = 24; /* Anfangswert */
    while(fusslaenge <= 38) { /* Scheifenbedingung */
        schuhgroesse_de();
        schuhgroesse_eu();
        schuhgroesse_usa();
        fusslaenge= fusslaenge +1; /* Inkrement */
    }
    /*oder */
    for(fusslaenge=24; fusslaenge<=38; fusslaenge= fuesslaenge+1) { /* Zaehlschleife */
        schuhgroesse_de();
        schuhgroesse_eu();
        schuhgroesse_usa();
    }
}
```

# 4. Schleifen

Schleifen ermöglichen die wiederholte Abarbeitung einer Anweisung oder eines unbenannten Blockes.

Berechne die 6! (die Fakultät von 6):  $6 * 5 * 4 * 3 * 2 * 1 * 1$

$0! = 1$ ,

die Fakultät von n berechnet sich aus der Fakultät von  $n * (n-1)!$  →  $n * (n-1) * (n-2) * \dots * 1 * 1$ ;

```
int fak = 1;
int n = 6;
while(n>0) {                               /* Schleifenbedingung */
    fak = fak * n;
    n = n-1;
}
printf("Die Fakultaet von %d ist %d",n, fak); /* 6! ist 720 */
```

# 4. While-Schleifen

While-Schleifen: „Tue etwas solange eine Bedingung gilt!“

```
while (y>=0){ /* Test am Anfang */  
    /* Alles wird solange ausgeführt, wie Bedingung erfüllt ist.  
    Möglicherweise nie, wenn Bedingung schon am Anfang nicht erfüllt .*/  
}  
  
do {  
    /* Alles wird mindestens einmal ausgeführt und solange die  
    Bedingung erfüllt ist */  
}while (y>=0); /* Test am Ende */
```

```
int erg =0;  
int i=0;  
while (i<6) {  
    erg= erg+i;  
    i=i+1;  
}
```

Bedingung erfüllt		Ja 0<6	Ja 1<6	Ja 2<6	Ja 3<6	Ja 4<6	Ja 5<6	nein 6<6
erg	0	0	1	3	6	10	15	
i	0	1	2	3	4	5	6	

# 4. For-Schleifen

For-Schleifen oder Zählschleifen haben generell folgenden Aufbau:

for(Anfangswerte; Endbedingungen; Schleifenendanweisungen) Anweisung oder Block

```
int i;  
for(i=0; i<=10; i++) /* 11 Schleifendurchläufe für i von 0 bis 10 */  
    printf(" i = %d\n",i);
```

Auch mehrere Anweisungen durch Komma getrennt sind möglich:

```
int i,j;  
for(i=0, j=20; i<=10; i++, j++) /* 11 Schleifendurchläufe für i */  
    printf(" i = %d, j= %d\n",i,j);
```

For und While-Anweisungen können häufig alternativ eingesetzt werden.

while	for
<pre>int erg =0; int i=0; while (i&lt;6) {     erg=erg+i;     i=i+1; }</pre>	<pre>int erg, i; for(i=0, erg=0; i&lt;6; i=i+1)     erg=erg+i;</pre>

# 4. Vorzeitige Beendigung von Schleifen

Enthält die Schleife einen Block, so können innerhalb des Blockes die Anweisungen `continue` und `break` genutzt werden.

- **continue** springt sofort zur Schleifenanweisung und ermöglicht so den vorzeitigen nächsten Durchlauf
- **break** springt aus dem Block und ermöglicht einen Abbruch der Schleife ohne Test auf Endbedingung;

```
int i, k;
for(i=0, k=1; i<1000; i=i+1) {
    if (k%42==0) break; /* wenn i durch 42 teilbar ist, beende Algorithmus */
    else if(i%100==0) continue; /* lasse volle Hunderter aus */
    else {
        k = k + i; // Addiere zu k der Wert von i
    }
    printf("i = %d, k=%d",i, k);
}
```

Wie weit kommt der Algorithmus?

# 4. Beispiele für For-Schleifen

## Vorsicht bei Schleifen mit Gleitkommazahlen und Test auf Gleichheit!

```
main() { float g;
    for(g=0; g<=1.0; g=g+0.1) /* ok*/
        printf(" g = %f\n",g);

    for(g=0;;g=g+0.1) {          /* ok ohne Endebedingung */
        printf(" g = %f\n",g);
        if (g>=1.0) break;
    }
    for(g=0;;) { /* ok ohne Ende und Inkrement */
        g=g+0.1;
        printf(" g = %f\n",g);
        if (g>=1.0) break;
    }
    g=0.0f;
    for(;;) { /* ok Ohne Initialisierung, Ende und Inkrement*/
        g=g+0.1;
        printf(" g = %f\n",g);
        if (g>=1.0) break;
    }
    for(g=0; g!=1.0; g=g+0.1) /*Fehler zum endlos, Rundung/Genauigkeitsproblem */
        printf(" g = %f\n",g);
}
```

➔ bei double und float-Werten im Vergleich == und != vermeiden, besser <= >=

# 5. Aufgaben

Summe und Mittelwert ( nächste Übung )

- Gib die Anzahl der Zahlen vor, dann in einer Schleife die Werte. Berechne dabei die Summe und den Durchschnitt.
- Gib die Werte ein und zähle die Anzahl der erfolgreichen Eingaben. falls die Zahl größer ist als 0, so bilde Summe und Mittelwert. Die Eingabe der Werte ist beendet, falls der Nutzer keine Zahl eingibt. (Tipp: Nutze die Bedingung  $((\text{scanf}(\text{„\%d“}, \&a) == 1))$ , da scanf die Anzahl der erfolgreichen Zahlenkonvertierungen zurückliefert.

Umrechnung Grad- in Bogenmaß

- Winkelangaben von Gradmaß (ganzzahlige Angaben) in Bogenmaß umwandeln ( Der Vollkreis hat  $360^\circ$  oder  $2 \cdot \pi$ )  $wb = \frac{2 \cdot \pi}{360} * wg$
- Bei Winkelangaben außerhalb von  $0..360$  gleichen Winkel im Kreis suchen!

Berechnung der Ziffern im Oktalsystem

- Bestimme die Ziffern einer Zahl im Oktalsystem. Die Reihenfolge der Ziffern kann umgedreht bestimmt werden (die kleinste zuerst).