4. Übung

1 Wiederholung Variable, Blöcke
2 Schleifen
3 Symbolische Konstanten
4 Aufgaben
  - Summe und Mittelwert
  - Erweiterungen der Kreisberechnung
  - Umrechnung Grad- in Bogenmaß
  - Ermittlung der größten enthaltenen Quadratzahl
  - Berechnung der Ziffern im Oktalsystem

1. Wiederholung Variable

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

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

```
int a; /* Variable a deklariert, definiert und nicht initialisiert */
a = 3; /* bestehende Variable 3 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)
```

1. 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);
}
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 */

}
```

1. Berechnung von Schuhgrößen

Schuhgröße (EU) = (*Fußlänge in* cm + 1,5) × 1,5
Schuhgröße (in Deutschland begräuchlich) = (*Fußlänge in* cm + 1,54) / 0.667

Brannock-System:
Herrengröße (US) = *Fußlänge in* cm ÷ 2,54 × 3 − 22
Damengröße (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();
}
```

1. Berechnung von 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();
}
```

1. 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 Inkrement.

```
…

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

}
```

1. Wiederholung

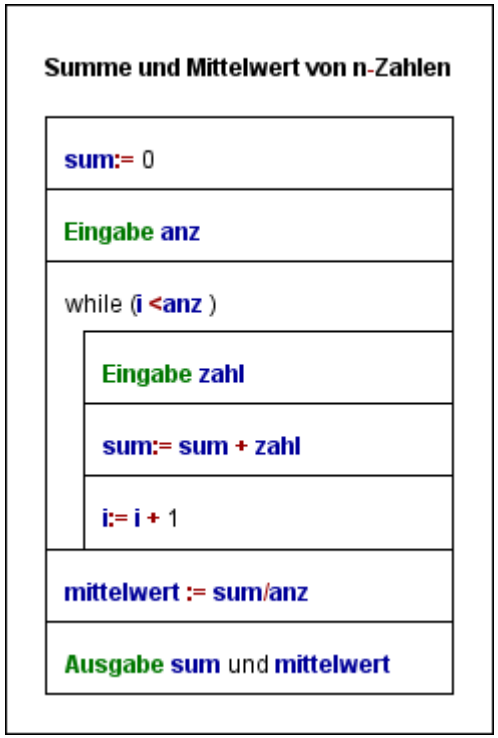
Wir wollen Schleifen in C - Programmen einsetzen:
Berechnung von Summe und Durchschnitt von n-Zahlen.

`Algorithmus?`

1. Wiederholung

Wir wollen Schleifen in C - Programmen einsetzen:
Berechnung von Summe und Durchschnitt von n-Zahlen.

Vorgabe: Wieviel Zahlen sollen eingegeben werden!
Wiederholte Eingabe einer Zahl, immer Zwischensummen bilden.
Letzte Zwischensumme ist Summe der zahl und Mittelwert ist Summe /Anzahl der zahlen.



1. Wiederholung

Wir wollen Schleifen in C - Programmen einsetzen:
Berechnung von Summe und Durchschnitt von n-Zahlen.

```
#include <stdio.h>
main(){
        int sum=0, anz, zahl, i = 0;
        double mittelwert;
        printf(" Bitte die Anzahl der Zahlen vorgeben");
        scanf("%d", &anz);  /* Anzahl der einzugebenden Werte vorgeben */
        while (i < anz) { /* Anzahl der eingegebenen Werte nicht erreicht ? */
          printf(" naechste Zahl");
          scanf("%d", &zahl);
          sum= sum+zahl; /* Berechnung Summe und Speicherung */
          i = i+1;       /* Erhoehung der Anzahl der eingegebenen Werte */
        }
        mittelwert = 1.0*sum/anz; /* Durchschnitt berechnen */
        printf(" Anzahl der Zahlen: %d, Summe: %d, Durchschnitt %f\n",anz,
                sum , mittelwert);
}
```

2. 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	

2. For-Schleifen

For-Scheifen oder Zählschleifen haben generell folgenden Aufbau:

for(Anfangswerte; Endbedingungen; Scheifenendanweisungen) Anweisung oder Block

```
int i;
for(i=0; i<=10; i++) /* 11 Schleifendurchläufe für i von 0 bis 10 */
         printf(" i = %d\n",i);
int j,k; /* Auch mehrere Anweisungen durch Komma getrennt sind möglich*/
for(j=0, k=20; j<=10; j++, k++) /* 11 Schleifendurchläufe für j*/
         printf(" i = %d, j= %d\n",i,j);
```

For und While-Anweisungen können häufig alternativ eingesetzt werden.

While	For

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

Enthält die For-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** spingt aus dem Block und ermöglicht einen Abbruch der Schleife ohne Test auf Endbedingung;

3. Symbolische Konstanten

Die Programmiersprache C ermöglicht die Nutzung **Symbolischer Konstanten**.
(An allen genutzten Stellen, wird der Wert der Konstante per Quelltext kopiert.
Notation:
#define Name Ersetzung
Beispiel: Berechnung des Umfangs und Flächeninhaltes eines Kreises aus dem Radius

```
Beispiele ohne symbolische Konstanten:
#include <stdio.h>

int main() { /* Berechnungen Kreis */

        float radius;      /* Speicherplatz fuer Radius */
        float umfang, flaeche;
        printf("Eingabe des Radius \n");
        scanf("%f",&radius);
        umfang = 2 * 3.14159265358979323846264338327 9 *radius;
        flaeche = 3.14159265358979323846264338327 9 *radius*radius;
        printf("Bei einem Radius von %f ist der Umfang %f, der Flaecheninhalt %f \n",
                radius, umfang, flaeche);
}

Ersetze Pi durch Variable oder symbolische Konstante
```

3. Symbolische Konstanten

Die Programmiersprache C ermöglicht die Nutzung **Symbolischer Konstanten**.
(An allen genutzten Stellen, wird der Wert der Konstante per Quelltext kopiert.
Notation:
#define Name Ersetzung
Beispiel: Berechnung des Umfangs und Flächeninhaltes eines Kreises aus dem Radius

```
Beispiele ohne symbolische Konstanten:
#include <stdio.h>
#define PI 3.14159265358979323846264338327 9
int main() { /* Berechnungen Kreis */

         float radius;      /* Speicherplatz fuer Radius */
         float umfang, flaeche;
         printf("Eingabe des Radius \n");
         scanf("%f",&radius);
         umfang = 2 * PI *radius;
         flaeche = PI *radius*radius;
         printf("Bei einem Radius von %f ist der Umfang %f, der Flaecheninhalt %f \n",
                  radius, umfang, flaeche);
}

Ersetze Pi durch Variable oder symbolische Konstante
```

3. Symbolische Konstanten

Die Programmiersprache C ermöglicht die Nutzung **Symbolischer Konstanten**.
(An allen genutzten Stellen, wird der Wert der Konstante per Quelltext kopiert.
Notation:
#define Name Ersetzung
Beispiel: Berechnung des Umfangs und Flächeninhaltes eines Kreises aus dem Radius

```
Beispiele ohne symbolische Konstanten:
#include <stdio.h>
#define PI 3.14159265358979323846264338327 9
int main() { /* Berechnungen Kreis */
          /* double pi = 3.14159265358979323846264338327 9; */ /* Variable ist aenderbar !*/
         float radius;      /* Speicherplatz fuer Radius */
         float umfang, flaeche;
         printf("Eingabe des Radius \n");
         scanf("%f",&radius);
         umfang = 2 * PI *radius;
         flaeche = PI *radius*radius;
         printf("Bei einem Radius von %f ist der Umfang %f, der Flaecheninhalt %f \n",
                  radius, umfang, flaeche);
}

Ersetze Pi durch Variable oder symbolische Konstante
```

4. Aufgaben

1. Summe und Mittelwert

- Summen und Mittelwertberechnung für die Eingabe von beliebig vielen Zahlen
- (Tipp: Nutze die Bedingung ((scanf(„%d“,&a)==1), da scanf die Anzahl der erfolgreichen Zahlenkonvertierungen zurückliefert.

2. Erweitere das Beispiel zur Kreisberechnung

- Wiederhole die Berechnung, solange eine Zahl für den Radius eingegeben wird.

3. Umwandlung von Winkelangaben in Gradmaß in Angaben in Bogenmaß

- Info: Ein Vollkreis hat 360° oder in Bogenmaß 6,1831 (2*PI). $wb = \frac{2*\pi}{360} * wg$
- Zu einem eingegebenen Winkel soll die Ausgabe in Bogenmaß erfolgen.
- Bei Winkelangaben außerhalb von 0..360 gleichen Winkel im Kreis suchen!
- Geben sie eine Tabelle von Umrechnungen von grad in Bogenmaß an.

4. Ermittle zu einer gegeben Zahl die größte enthaltene Quadratzahl

Beispiel Eingabe: 42, Antwort 36, der Abstand ist 6.

Bitte Zahl eingeben: 19, Antwort 16, der Abstand ist 3.

5. Berechnung der Ziffern im Oktalsystem

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