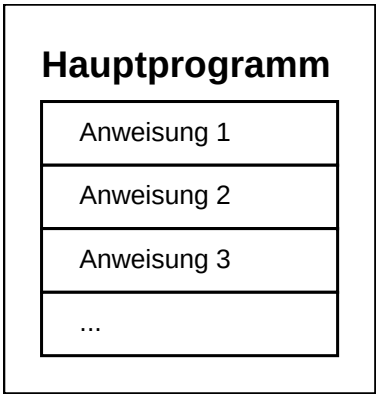
Grundlegende Programmstrukturen sind Sequenzen, Schleifen und Entscheidungen.

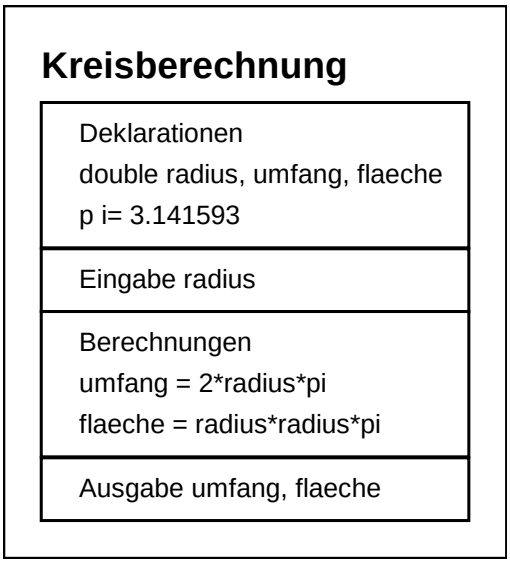
Sequenzen

Eine Sequenz ist eine Folge von Anweisungen (Anweisungsblock) ohne Verzweigungen. Der Programmablauf ist linear von oben nach unten. Ein Anweisungsblock befindet sich normalerweise in geschweiften Klammern { … }.



Typische Beispiele sind Programme, welche nach dem EVA-Prinzip organisiert sind (Eingabe-Verarbeitung-Ausgabe).

Beispiel: Kreisberechnung

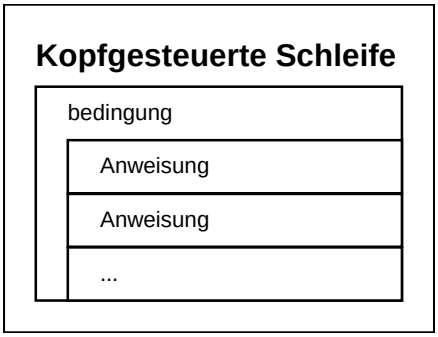


```
#include <stdio.h>
#include <conio.h>
main()
{                                          //Beginn des Anweisungsblocks
      double radius=0,umfang=0,flaeche=0;  //Deklarationen
      const double pi=3.141593;
      printf("\nKreisberechnung\n");
      printf("\nRadius: ");                //Eingabe
      scanf("%lf",&radius);
      umfang=2*radius*pi;                  //Berechnungen
      flaeche=radius*radius*pi;
      printf("\nUmfang : %8.3lf",umfang);  //Ausgabe
      printf("\nFlaeche: %8.3lf",flaeche);
      return(0);
}                                          //Ende des Anweisungsblocks
```

Schleifen

Schleifen bestehen aus einem Anweisungsblock (Schleifenrumpf) und einer Ausführungsbedingung. Der Schleifenrumpf wird solange wiederholt, wie die Ausführungsbedingung erfüllt ist. Je nachdem, zu welcher Zeit die Bedingung abgeprüft wird, unterscheidet man zwischen kopf- und fußgesteuerten Schleifen.

Kopfgesteuerte Schleife:



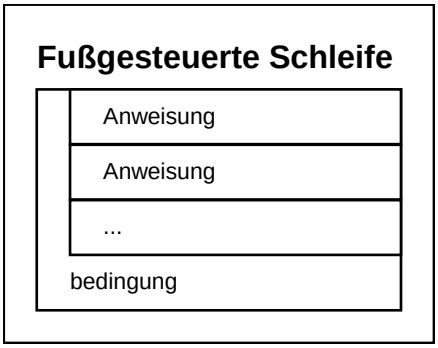
Syntax:

```
while(bedingung)
      {
      anweisung;
      anweisung;
      …;
      }
```

Wird nach der while-Bedingung nur eine Anweisung ausgeführt, kann der Anweisungsblock mit { } entfallen:

while(bedingung) anweisung;

Fußgesteuerte Schleife:



Syntax:

```
do
      {
      anweisung;
      anweisung;
      …;
      }
while(bedingung);
```

Fußgesteuerte Schleifen finden Verwendung, wenn von der Aufgabenstellung her der Schleifenrumpf mindestens einmal ausgeführt werden muss.

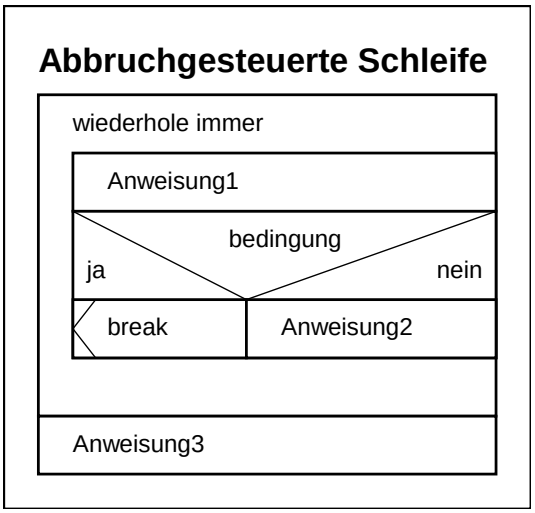
Auch hier kann der Anweisungsblock mit { } entfallen, wenn der Schleifenrumpf nur eine Anweisung enthält:

```
do anweisung;
while(bedingung);
```

Auf jeden Fall ist zu beachten, dass bei fußgesteuerten Schleifen die `while`-Bedingung mit einem Semikolon (;) abschließen muss!

Abbruchgesteuerte Schleife:

Gelegentlich ergibt sich die Abbruchbedingung während des Schleifenrumpfes (z.B. auf Grund einer Eingabe). Mit dem Schlüsselwort `break` kann ein Schleifenrumpf verlassen werden. Das Programm wird mit der nach dem Schleifenrumpf folgenden Anweisung fortgesetzt.



Syntax:

```
…
while(1)
{
anweisung1;
if(bedingung) break;
else anweisung2;
}
anweisung3;
…
```

Die Anweisung `while(1)` bewirkt, dass die Schleifenbedingung immer erfüllt ist. Ein Abbruch erfolgt erst nach der `if`-Bedingung (s.u.).

For-Schleifen:

Bei vielen Schleifen wird die Anzahl der Durchläufe durch einer Zähler bestimmt (z.B. der Index bei einem Datenfeld). Der Zähler beginnt mit einem Anfangswert und wird bei jedem Schleifendurchlauf erhöht/erniedrigt bis ein bestimmter Endwert erreicht wird.

Syntax:

```
for (initialisierung;bedingung;inkrementausdruck)
      {
      anweisung;
      anweisung;
      …;
}
```

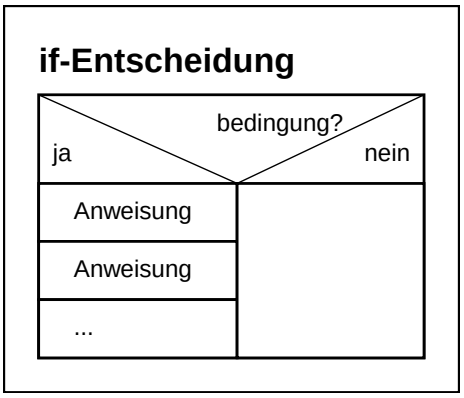
Durch die Initialisierung wird der Anfangswert des Zählers festgelegt. Der Bedingungsausdruck legt die Ausführungsbedingung fest. Der Inkrementausdruck bestimmt, um welchen Wert sich der Zähler nach einem Durchlauf ändert. Der Inkrementausdruck kann auch negativ sein. Der Zählwert wird dann entsprechend verkleinert.

Das folgende Beispiel schreibt die Zahlen 1 bis 10 und die zugehörigen Quadratzahlen auf den Bildschirm:

```
#include <stdio.h>
main()
{
      int i=0;
      for(i=1;i<=10;i++)
      {
            printf("\nZahl: %2i\tQuadrat: %3i",i,i*i);
      }
      return(0);
}
```

Entscheidungen

Mit dem Schlüsselwort `if` wird eine bedingte Entscheidung eingeleitet. Der folgende Anweisungsblock wird nur ausgeführt, wenn eine eindeutige Bedingung erfüllt ist. Andernfalls werden die Anweisungen ausgeführt, welche sich in einem `else`-Block befinden. Der `else`-Block kann auch entfallen.

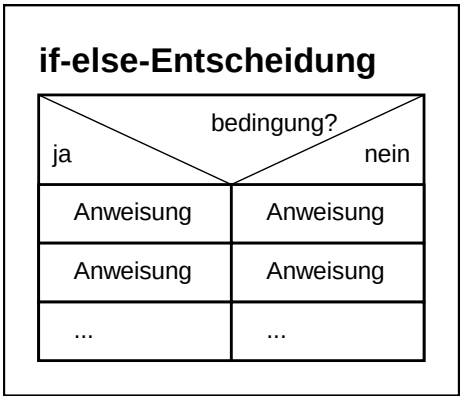


Syntax:

```
if (bedingung)
      {
      anweisung;
      anweisung;
      …;
      }
```

Wird nach `if (` oder `else)` nur eine einzelne Anweisung ausgeführt können die { }-Klammern entfallen.

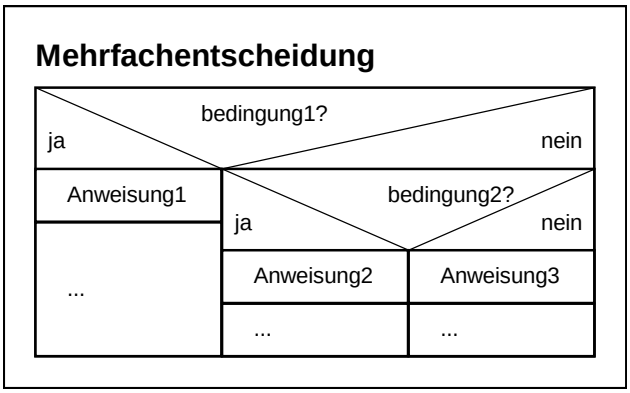
```
if (bedingung) anweisung;
```



Syntax:

```
if (bedingung)
      {
      anweisung;
      anweisung;
      …;
      }
else
      {
      anweisung;
      anweisung;
      …;
      }
```

if-else-Anweisungen können beliebig verschachtelt werden:



Syntax:

```
if (bedingung1)
      {
      anweisung1;
      …;
      }
else if (bedingung2)
      {
      anweisung2;
      …;
      }
      else
      {
      anweisung3;
      …;
      }
```

Bedingungsoperatoren

Die Bedingungen bei Schleifen und Entscheidungen ergeben sich entweder aus der Auswertung eines einzelnen Ausdruckes oder aus dem Vergleich zweier Werte.

Eine Bedingung ist solange erfüllt, wie der Ausdruck einen Wert ungleich 0 liefert.

Beispiele:

```
while(1)
      {
      …;
      }
```

Die Bedingung ist immer erfüllt, weil der Bedingungsausdruck immer 1 ist. Der Abbruch der Schleife ist nur mit einer break-Anweisung möglich.

Im folgenden Beispiel wird 5 mal der Text ‚Hallo' ausgegeben, bis die Variable i den Wert 0 annimmt:

```
#include <stdio.h>
main()
{
      int i=5;
      while(i)
      {
            printf("\nHallo");
            i--;
      }
      return(0);
}
```

Für Bedingungen, die auf den Vergleich zweier Werte beruhen, gelten folgende Bedingungsoperatoren:

```
==    gleich
!=    ungleich
>     größer als
<     kleiner als
>=    größer oder gleich
<=    kleiner oder gleich
```

Beispiele:

```
while(i<10)           //solange i < 10
if (zahl1==zahl2)     //wenn zahl1 gleich zahl2, ein einfaches '=' wäre eine
                      //Zuweisung!
```

Logische Operatoren erlauben mehrere Bedingungen gleichzeitig abzuprüfen:

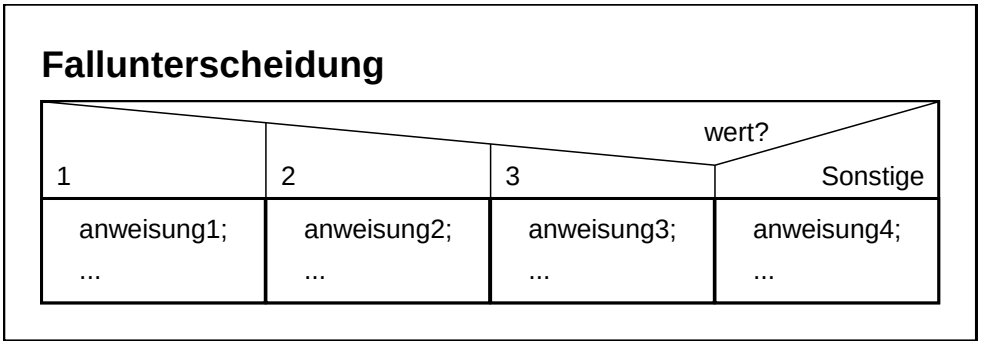
```
&&   logisches UND
||   logisches ODER

if((zahl1>0) && (zahl1<=10))    //zahl1 größer 0 und kleiner gleich 10
while((x==0x41) || (x==0x61))   //solange x=41H oder 61H
```

Fallunterscheidung

Verschachtelte if-Anweisungen mit vielen Bedingungen sind häufig unübersichtlich. Vielfach bietet sich die Technik der Fallunterscheidung mit der **switch-case-Anweisung** an. Dabei wird in Abhängigkeit von einem Zahlenwert zwischen mehreren Fällen unterschieden. Jeder Fall besitzt einen eigenen Anweisungsblock.

Struktogramm:



Syntax:

```
switch(wert)          //wert ist eine Variable vom Typ Integer oder char
{
      case 1:
            anweisung1;
            …;
            break; //Entscheidungsblock beenden
      case 2:
            anweisung2;
            …;
            break;
      case 3:
            anweisung3;
            …;
            break;
      default:
            anweisung4;
            …;
}
```

Wirkungsweise:

Die Variable *wert* ist eine ganze Zahl (int oder char). Bei Ausführung des Blockes werden die einzelnen Fälle (case-Blöcke) von oben beginnend überprüft. Bei erfüllter Bedingung werden alle Anweisungen ausgeführt bis die **break**-Anweisung erreicht wird. Trifft keine Bedingung zu, werden die Anweisungen im `default`-Block durchgeführt (z.B.: eine Fehlermeldung). Der `default`-Block kann entfallen.

Die break-Anweisungen sind erforderlich, um den Anweisungsblock zu verlassen. Andernfalls werden auch die folgenden Anweisungsblöcke ausgeführt. Das kann benutzt werden, um mehrere Bedingungen abzuprüfen.

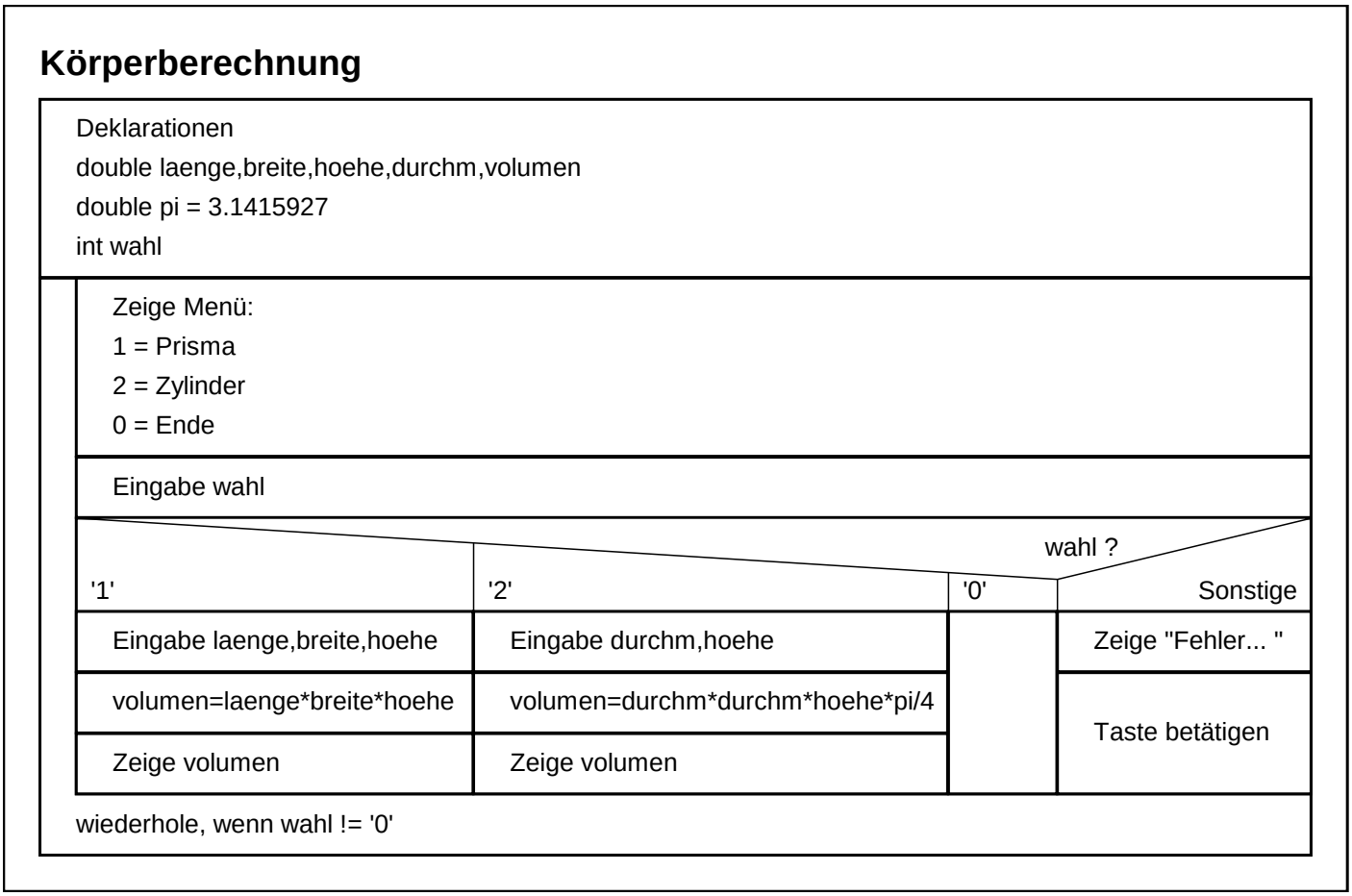
Beispiel:

Die folgende case-Anweisung überprüft, ob die Variable **`wert`** 'j' oder 'J' ist[1]:

```
switch(wert)
{
case 'j':
case 'J':
      anweisungen…;break;
…
}
```

Häufig werden Fallunterscheidungen angewendet, um eine Menüauswahl zu treffen. Das folgende Beispiel gestattet aus einem Menü heraus zu wählen, ob das Volumen eines Prismas (Menüpunkt = 1) oder eines Zylinders (Menüpunkt = 2) berechnet werden soll. Bei Eingabe des Menüpunktes = 0 wird das Programm beendet. Werden andere Werte eingegeben wird eine Fehlermeldung erzeugt.

Struktogramm:



[1] 'J' bzw. 'j' repräsentieren den ASCII-Code des jeweiligen Zeichens

```
//Programm zur Koerperberechnung
#include<stdio.h>
#include<conio.h>
main()
{
      double laenge,breite,hoehe,durchm,volumen;
      const double pi=3.1415927;
      int wahl=0;
      do
      {
            printf("\nKoerperberechnung\n");
            printf("\n1 = Prisma");
            printf("\n2 = Zylinder");
            printf("\n\n0 = Ende");
            printf("\n\nBitte waehlen... ");
            wahl=getch();
            switch(wahl)
            {
            case '1':
                        printf("\nPrismaberechnung\n");
                        printf("\nLaenge: ");scanf("%lf",&laenge);
                        printf("\nBreite: ");scanf("%lf",&breite);
                        printf("\nHoehe : ");scanf("%lf",&hoehe);
                        volumen=laenge*breite*hoehe;
                        printf("\nVolumen = %8.3lf\n",volumen);
                        break;
            case '2':
                        printf("\nZylinderberechnung\n");
                        printf("\nDurchmesser: ");scanf("%lf",&durchm);
                        printf("\nHoehe      : ");scanf("%lf",&hoehe);
                        volumen=durchm*durchm*hoehe*pi/4;
                        printf("\nVolumen = %8.3lf\n",volumen);
                        break;
            case '0':
                        break;
            default:
                        printf("\n\nFalsche Eingabe... ");getch();
            }
      }
      while(wahl != '0');
      return(0);
}
```

Übung: Körperberechnung

Ergänzen Sie das obige Programm gemäß folgender Vorgaben:

- Das Menü soll um einen Punkt zur Berechnung einer Kugel erweitert werden (Menüpunkt = 3).
- Außerdem soll zu allen Körpern zusätzlich die Oberfläche berechnet werden.

Übung : Zinseszinsrechnung

Entwickeln Sie ein Programm, welches die Kapitalentwicklung bei Zinseszinsen berechnet.

Eingabedaten:

Anfangskapital k0
Zinsfuß p in %
Laufzeit j in Jahren

Die formatierte Ausgabe soll für jedes einzelne abgelaufene Jahr die Zinsen und das Endkapital anzeigen. Bei Betätigung der j-Taste soll eine neue Berechnung möglich sein.

Beispiel für Konsolenanzeige:

```
Anfangskapital: 100
Zinsfuss in %: 10
Laufzeit in Jahren: 4

1.Jahr  Zinsen: 10.00  Endkapital:  110.00
2.Jahr  Zinsen: 11.00  Endkapital:  121.00
3.Jahr  Zinsen: 12.10  Endkapital:  133.10
4.Jahr  Zinsen: 13.31  Endkapital:  146.41

Nochmal? j/n
```

Entwickeln Sie ein **Struktogramm** und ein **C-Programm**!

Übung : Zahlenraten

Entwickeln Sie ein Programm, welches eine Zufallszahl von 1 bis 999 erzeugt. Die Zahl soll von dem Benutzer erraten werden.

Dazu gibt der Benutzer eine Zahl ein. Wenn die eingegebene Zahl größer als die Zufallszahl ist, meldet der Rechner 'Die Zahl ist zu groß!'. Wenn die eingegebene Zahl kleiner als die Zufallszahl ist, meldet der Rechner 'Die Zahl ist zu klein!'. Die Eingabe wird solange wiederholt, bis der Rechner 'Zahl gefunden!' ausgibt.

Die Anzahl der Versuche soll gezählt werden und danach ausgegeben werden.

Zusatz:

In Abhängigkeit von der Anzahl der Versuche soll folgende Bewertung ausgegeben werden:

Anzahl < 10 --> 'sehr gut'
11 < Anzahl < 13 --> 'gut'
14 < Anzahl < 15 --> 'na ja...'
Anzahl > 15 --> 'das üben wir nochmal...'