4. Kontrollstrukturen

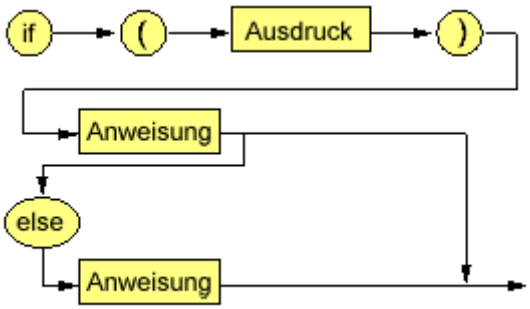
4.1. Linearer Ablauf von Programmen

C Programmen sind dadurch gekennzeichnet, dass alle Befehle welche in der Funktion main( )stehen linear d.h. ein Befehl nach dem anderen ausgeführt wird.

Das Prinzip welches dahinter steht nennt sich EVA und sagt nichts anderes aus das zuerst ein Programm Eingaben erwartet – von wo diese auch immer kommen, diese Eingaben dann Verarbeitet werden und zum Schluss die Ausgabe des Ergebnisses erfolgt.

4.2. Auswahlstrukturen mit if .. else

Die Auswahlstruktur ist gekennzeichnet durch einen nicht linearen Ablauf mit einer Vorwärtsverzweigung. Der Ablauf gelangt an einen Entscheidungspunkt, an dem, abhängig von einem Bedingungsausdruck, unterschiedliche Verarbeitungswege eingeschlagen werden können. Der Bedingungsausdruck ist ein logischer (boolescher) Ausdruck, dessen Ergebnis ein Wahrheitswert (wahr oder falsch) ist.



einseitige Auswahl

Diese Alternativstruktur führt nur auf einem der beiden Verzweigungspfaden eine Anweisungsfolge aus und endet in der Zusammenführung beider Pfade.

```
if (Bedingungsausdruck) Anweisung;
```

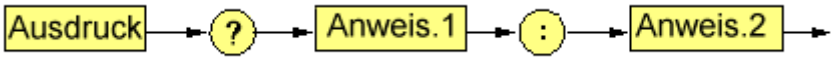
zweiseitige Auswahl

Bei dieser Alternativstruktur führt jeder Verzweigungspfad auf jeweils eine eigene Anweisungsfolge. Sie endet auch wieder in einer Zusammenführung der Pfade.

```
if  (Bedingungsausdruck)
   Anweisung1;
else
   Anweisung2;
```

4.3. Bedingungsausdrücke mit dem Operator ?:

Die Syntax eines Ausdrucks mit diesem Operator hat folgendes Aussehen:



Ein Ausdruck mit dem Bedingungsoperator kann nicht alleine stehen (wie if ...), sondern innerhalb eines Ausdrucks (z. B. einer Wertzuweisung). Ist das Ergebnis des Bedingungsausdrucks wahr (!= 0), wird Ausdruck1 verwendet, sonst Ausdruck2.

Beispiele: Vorzeichenoperator (Zahl negativ: vz = -1, Zahl positiv oder Null: vz = +1) und Suche nach dem Maximum:

vz = (Zahl < 0) ? -1 : +1
max = (a > b) ? a : b

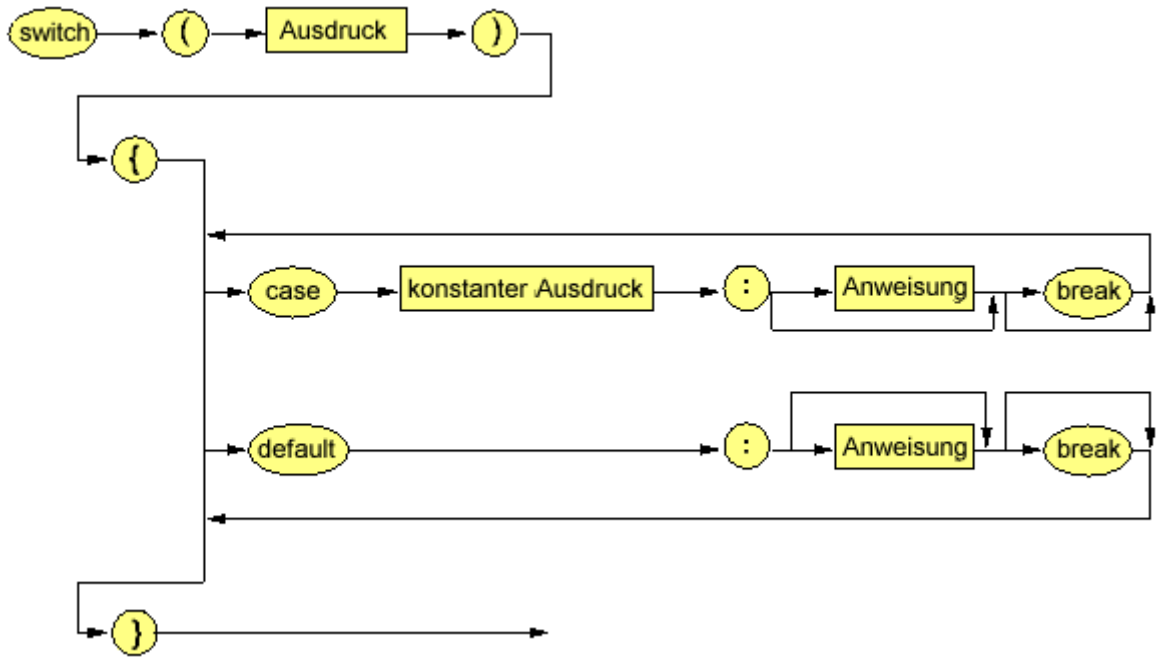
4.4. Bedingungsausdrücke mit if …. elseif ….. else

```
#include <stdio.h>

void main(void)
{
  char operator;
  int  wert1, wert2, erg ;

  if (scanf("%d %c %d",&wert1, &operator, &wert2) == 3)
    {
    if      (operator== '+')
              erg = wert1 + wert2;
    else if (operator == '-')
              erg = wert1- wert2;
    else if (operator=='*')
              erg = wert1 * wert2;
    else if (operator=='/')
              erg = wert1 / wert2;
    else    printf("Falsche Eingabe\n");
    printf("%d %c %d=%d\n",wert1,operator,wert2,erg);
    }
  else
    printf("zu wenig Eingabewerte\n");
}
```

4.5. Mehrfachauswahl mit switch .. case

Bei dieser Struktur gibt es mehr als zwei Auswahlpfade, die aus einer Verzweigung ihren Ausgang nehmen und in einer Zusammenführung enden. Hier erfolgt die

Abfrage nicht nach einem Bedingungsausdruck, sondern der bedingte Ausdruck liefert einen Wert. Für jeden möglichen Ergebniswert ist ein Zweig vorgesehen. Existiert nicht für jeden möglichen Ergebniswert der Bedingung ein Pfad, ist ein zusätzlicher Pfad für alle nicht behandelten Fälle vorzusehen, der default Zweig.

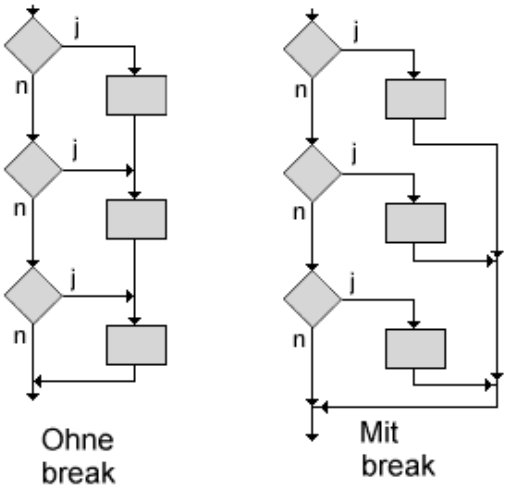


Wenn der Ausdruck den Wert W_i besitzt, wird die Anweisungsfolge $Anweisung_i$ und alle folgenden ausgeführt. Will man das vermeiden, muss der jeweilige switch -Zweig durch break; abgeschlossen werden. Das sieht dann so aus:

switch (Ausdruck)
  {
  case W_1: $Anweisung_1$; break;
  case W_2: $Anweisung_2$; break;
          ...; break;
  case W_n: $Anweisung_n$; break;
  default: $Anweisung_{def}$;

  }

Den Unterschied machen auch die beiden folgenden Ablaufdiagramme sichtbar:

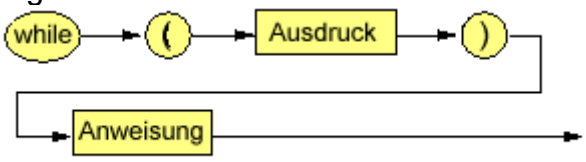


4.6. Wiederholungsstrukturen mit while

Wiederholungsstrukturen ergeben sich, wenn eine Anweisungsfolge zur Lösung einer Aufgabe mehrfach durchlaufen werden soll (z. B. das Bearbeiten aller Komponenten eines Feldes). Es liegt ein nichtlinearer Verlauf mit Rückwärtsverzweigung vor. Die Programmierung einer Wiederholungsstruktur führt zu einer so genannten "Programmschleife". Wichtig ist die Terminierung der Schleife, d. h. mindestens eine Anweisung muss dafür sorgen, dass eine Variable derart verändert wird, dass nach einer endlichen Zahl von Durchläufen die Abfragebedingung nicht mehr erfüllt ist. Ist dies nicht gegeben spricht man von einer so genannten Endlosschleife.

Abweisende Wiederholung

In diesem Fall steht die Abfrage zu Beginn der Schleife (also vor der Anweisungsfolge). Ist der Bedingungsausdruck schon beim Eintritt in die Anweisungsstruktur nicht erfüllt, wird die Anweisungsfolge überhaupt nicht ausgeführt.

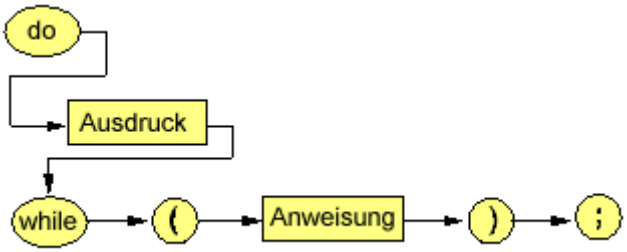



```
while (Bedingungsausdruck)
  Anweisung;
```

4.7. Wiederholungsstrukturen mit do .. while

Nicht abweisende Wiederholung

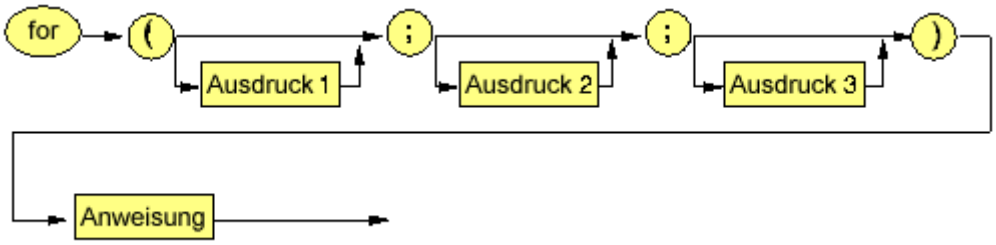
In diesem Fall steht die Abfrage am Ende der Schleife also nach der Anweisungsfolge. Die Anweisungsfolge wird auf jeden Fall mindestens einmal ausgeführt.




```
do
  Anweisung;
while (Bedingungsausdruck);
```

4.8. Wiederholungsstrukturen mit for

Diese Anweisungsstruktur stellt die häufigste Form der abweisenden Wiederholung dar.



Ausdruck1:
Der Anfangsausdruck wird einmal zu Beginn vor dem Durchlaufen der Schleife ausgeführt; er kann z.B. zum Vorbesetzen einer Zählvariablen benutzt werden.
Ausdruck2:
Darauf folgt eine abweisende Wiederholung, analog den Regeln der while-Schleife, d.h. der Bedingungsausdruck wird ausgewertet und nur bei erfüllter Bedingung der Anweisungsteil durchlaufen.
Ausdruck3:
Zuletzt wird der Veränderungsausdruck bearbeitet, z. B. kann hier die Zählvariable inkrementiert werden, die zu Beginn des nun folgenden Schleifendurchlaufs im Bedingungsausdruck geprüft wird, um ggf. die Schleife abzubrechen.

```
for(init; bedingung; inkrement)
  anweisung;

entspricht

init;
while(bedingung)
  {
  anweisung;
  inkrement;
  }
```

4.9. Unstrukturierte Kontrollanweisungen

In C ist es möglich, beliebige Anweisungen durch eine Marke zu kennzeichnen. Die Marke folgt dabei den Namenskonventionen für Variablen. Die Marke wird mit einen Doppelpunkt abgeschlossen.

```
Marke:
```

Diese Marke kann mit dem `goto` Befehl angesprungen werden.

```
goto Marke;
```

Sprungbefehle sollten nie verwendet werden.
Daneben gibt es noch zwei weitere unstrukturierte Verzweigungsbefehle:
break bricht die Bearbeitung der aktuellen strukturierten Anweisung ab.
continue führt einen Sprung zum Schleifen Ende aus, um dann mit dem nächsten Schleifendurchlauf fortzufahren.