

Übersicht

Lehrinhalt: Programmierung in C

- Überblick über Programmiersprachen, Allgemeines
- C-Basisdatentypen, Zahlendarstellung, Variablen, Konstanten
- Operatoren und Ausdrücke
- **Anweisungen**
- **Kontrollstrukturen**
- Funktionen
- Zeiger und Felder
- Zeichenketten (Strings)
- Benutzerdefinierte Datentypen
- Dynamischer Speicher
- Dateiarbeit
- Funktionspointer, Rekursion
- Preprozessor

Anweisungen und Kontrollstrukturen

Anweisungen werden im Programm nacheinander als Sequenz abgearbeitet, wenn nichts anderes angegeben ist.

Einzelne Anweisung mit einer Zuweisung, zum Beispiel:

```
A = 2*r*r;
```

Eine Anweisung ist auch der Aufruf einer Funktion, zum Beispiel:

```
printf("Der Erdumfang betraegt %lf km\n", 2.0*3.14159*erdradius);
```

Verbundanweisungen (auch als Blöcke bezeichnet) mit lokalen Variablen:

```
{ int x=1, y=2, z;  
  z=2*x+4*y;  
  printf("%d %d %d\n", x,y,z);  
}
```

Kontrollstrukturen

Beeinflussung des Kontrollflusses, d.h. der Reihenfolge, wie die Anweisungen abgearbeitet werden.

1. Sequenz

2. Selektion (auch Alternative)

zwingend
notwendig

3. Zyklus

4. Funktionsaufruf (Unterprogramme)

und Varianten von 3,4

Selektion: if

Die if-Anweisung ist das Ausdrucksmittel für die Selektion in C.

if (Bedingung) Anw;

*if (Bedingung)
Verbundanweisung*

Beispiel:

```
int monat;  
printf("Bitte Monat Ihres Geburtsdatums eingeben:");  
scanf("%d",&monat);  
if (monat<1 || monat>12)  
{ printf("falscher Wert für Monat eingegeben!\n");  
  exit(EXIT_FAILURE);  
}  
...
```

Selektion: if-else

```
if ( Bedingung ) Anw1;  
else Anw2;
```

```
if (Bedingung)  
    Verbundanweisung1  
else  
    Verbundanweisung2
```

Beispiel:

```
char kuehlen = 0, heizen = 0;  
...  
if (temperatur-zielwert > 2)  
{ printf("zu warm -> kuehlen\n");  
  kuehlen = 1;  
}  
else {  
  if (zielwert - temperatur > 2 )  
  { printf("zu kalt -> heizen\n");  
    heizen = 1;  
  }  
}  
} Peter Sobe  
...
```

Mehrfachselektion: switch-case

Zur Selektion unter mehreren alternativen Zweigen

```
switch(Fallausdruck)  
{  
  case Wert_1:  
    Anw_1; break;  
  case Wert_2:  
    Anw_2; break;  
  ...  
  case Wert_n:  
    Anw_n; break;  
  
  default: // kann entfallen  
    Anw_0;  
}
```

Mehrfachselektion: switch-case

Beispiel

```
unsigned int tag, monat, jahr;  
... // Eingabe des Datums  
switch(monat)  
{  
    case 1:  
        printf("Januar"); break;  
    case 2:  
        printf("Februar"); break;  
    case 3:  
        printf("Maerz");break;  
    ...  
    default:  
        printf("-undefiniert-");  
}
```

In einen Fall (case) wird bei entsprechendem Wert (Konstante) des Fallausdrucks gesprungen.

Nach case folgt eine oder mehrere Anweisungen. Achtung keine Verbundanweisung mit {...}

Die *break-Anweisung* ist erforderlich, damit nicht zusätzlich der jeweils folgende Fall auch abgehandelt wird!

Zyklus: while

Allgemeine Formen:

while (Bedingung) Anweisung;

*while (Bedingung)
Verbundanweisung*

- Die Anweisung oder Verbundanweisung wird solange wiederholt ausgeführt, wie die Bedingung zutrifft.
- Durch Änderungen der Variablenwerte wird die Bedingung i.d.R. nach endlich vielen Durchläufen irgendwann nicht mehr zutreffen und die Wiederholung endet.
- Trifft die Bedingung bei Eintritt in den Zyklus nicht zu, wird die Anweisung nicht (auch nicht ein einziges mal) ausgeführt.

Zyklus: while

Beispiel:

```
int summe = 0;
```

```
int i = 10;
```

```
while (i >= 1)
```

```
{ // entspricht while (i>0), entspricht while (i)
```

```
    summe = summe + i;
```

```
    i = i - 1;
```

```
}
```

```
printf("Die Summe der Zahlen von 1 bis 10 ist %d \n", summe);
```

```
....
```

Zyklus: do-while

Allgemeine Formen:

***do** Anweisung; **while** (Bedingung);*

***do** Verbundanweisung
while (Bedingung);*

- Der do-while-Zyklus wird mindestens einmal durchlaufen.
- Die Anweisung oder Verbundanweisung wird solange noch einmal ausgeführt, wie die Bedingung nach der Ausführung zutrifft.
- Typischerweise ändert die Anweisung oder Verbundanweisung den Inhalt der Variable, die für die Bedingung benutzt werden. Damit wird erreicht, dass die Wiederholung irgendwann endet.

Zyklus: do-while

Beispiel:

```
int summe = 0;  
int i = 10;
```

```
do {  
    summe = summe + i;  
    i = i - 1;  
} while (i >= 1);
```

```
printf("Die Summe der Zahlen von 1 bis 10 ist %d \n",summe);  
...
```

Zähl-Zyklus: for

Spezielle Form für einen Zyklus, die alle zur Zyklensteuerung benutzten Anweisungen und Ausdrücke zusammenfasst:

for (Lv=anfangswert ; Lv<=endwert ; Lv=Lv+schrittwert)
Anweisung;

Lv dient hier als Zählvariable

Allgemeine Formen:

for (Init-Anweisung ; Bedingungs-Ausdruck ; Schritt-Anweisung)
Anweisung;

for (Init-Anweisung ; Bedingungs-Ausdruck ; Schritt-Anweisung)
Verbundanweisung

Solange die Bedingung **erfüllt** ist, wird die Anweisung wiederholt.
Die Ausdrücke in der for (; ;) - Klammerung können auch leer sein.

Zähl-Zyklus: for

Beispiel 1:

```
int i;  
int summe = 0;  
for (i = 10; i > 0; i=i-1)  
    summe = summe + i;  
  
printf("Summe: %d \n",summe);  
...
```

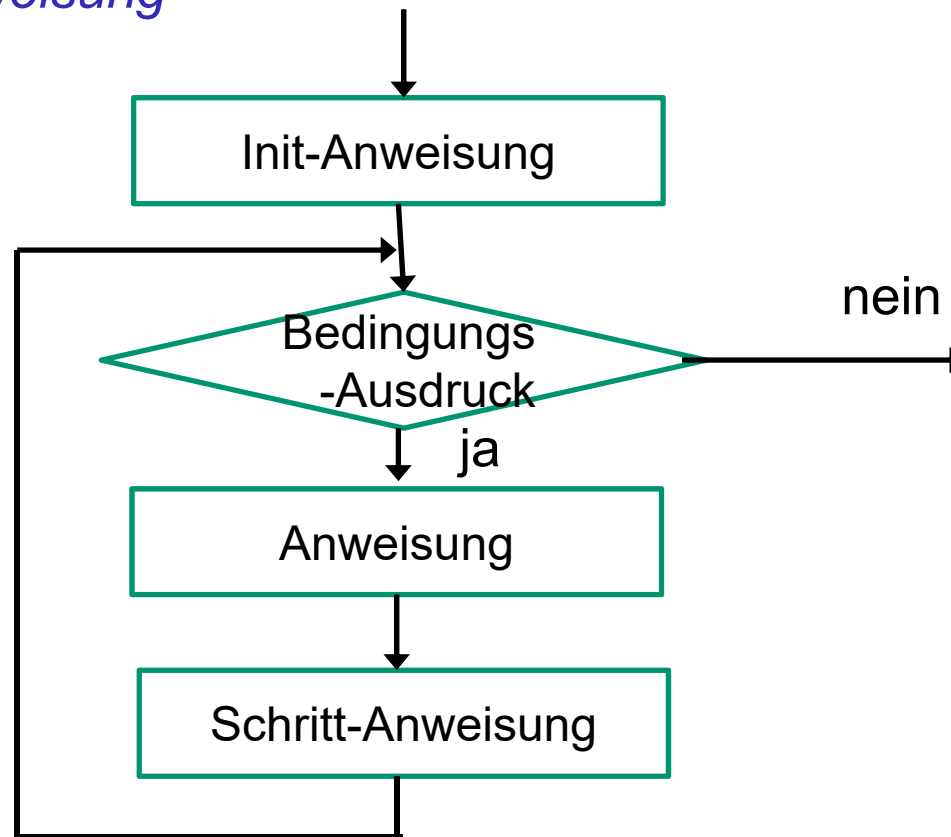
Beispiel 2:

```
int i;  
int summe = 0, produkt=1;  
for (i = 10; i > 0; i=i-1)  
{ summe = summe + i;  
  produkt = produkt * i;  
}  
printf("Summe: %d \n",summe);  
printf("Produkt: %d \n",produkt);  
  
...
```

Zähl-Zyklus: for

Wirkungsweise des for-Zyklus als Programmablaufplan:

*for (Init-Anweisung ; Bedingungs-Ausdruck ; Schritt-Anweisung)
Anweisung*



Zähl-Zyklus: for

Beispiel 3, Zwerge durchzählen:

```
int zwergnr;  
for ( zwergnr=1 ; zwergnr<=7 ; zwergnr=zwergnr+1 )  
printf("Zwerg Nr. %d meldet sich zur Arbeit. \n", zwergnr);
```

Beispiel 4, Scannen von Termineinträgen:

```
int h,m; // h ... Stunden, m ... Minuten  
for (h=0; h<24;h=h+1)  
{ for (m=0; m<60; m=m+1)  
{  
    if ( termin(h,m) ) printf("Termin um %02d:%02d \n",h,m);  
}  
}
```

Wir nehmen an, dass termin() eine Funktion ist, die 0 oder 1 zurückgibt

break-Anweisung

Die *break*-Anweisung kann in allen Zyklen verwendet werden, um die aktuelle Iteration vorzeitig zu beenden, d.h. vor Erreichen der normalen Ende-Bedingung. Es wird danach der Zyklus verlassen.

Anwendung für: while-, do-while- und for-Zyklen

Beispiel:

```
for (i=0; i<imax; i++)  
{ if (bedingung1) { break; }  
  ...  
  if (bedingung2) { break; }  
  ...  
  if (bedingung3) { break; }  
  ...  
}
```


continue-Anweisung

Die *continue*-Anweisung kann in allen Zyklen verwendet werden, um die aktuelle Ausführung des Zyklenkörpers vorzeitig zu beenden, d.h. vor Erreichen der normalen Ende-Bedingung. Es wird dann mit der folgenden Iteration fortgesetzt.

Anwendung für: while-, do-while- und for-Zyklen

Beispiel:

```
for (i=0; i<imax; i++)  
{  
    // do something  
    ...  
    if (bedingung) { continue; }  
    ...  
    // do something else  
    ...  
}
```

Nutzung der verschiedenen Zyklen

Die drei verschiedenen Varianten für den Zyklus – while, do-while und for – können wahlweise verwendet werden.

Aspekte für eine sinnvolle Auswahl:

- Verwendung von **while**, wenn die Anzahl der Iterationen n nicht vorab bekannt ist, mit $n \geq 0$ (auch null Durchläufe möglich!)
- Verwendung von **do-while**, wenn im Gegensatz dazu $n \geq 1$ (mindestens ein Durchlauf!) beabsichtigt ist.
- und Bevorzugung des **for-Zyklus** dann, wenn die Anzahl der Iterationen beim Eintritt in den Zyklus bekannt ist.