

Programmiersprache 1 (C++)

Prof. Dr. Stefan Enderle
NTA Isny

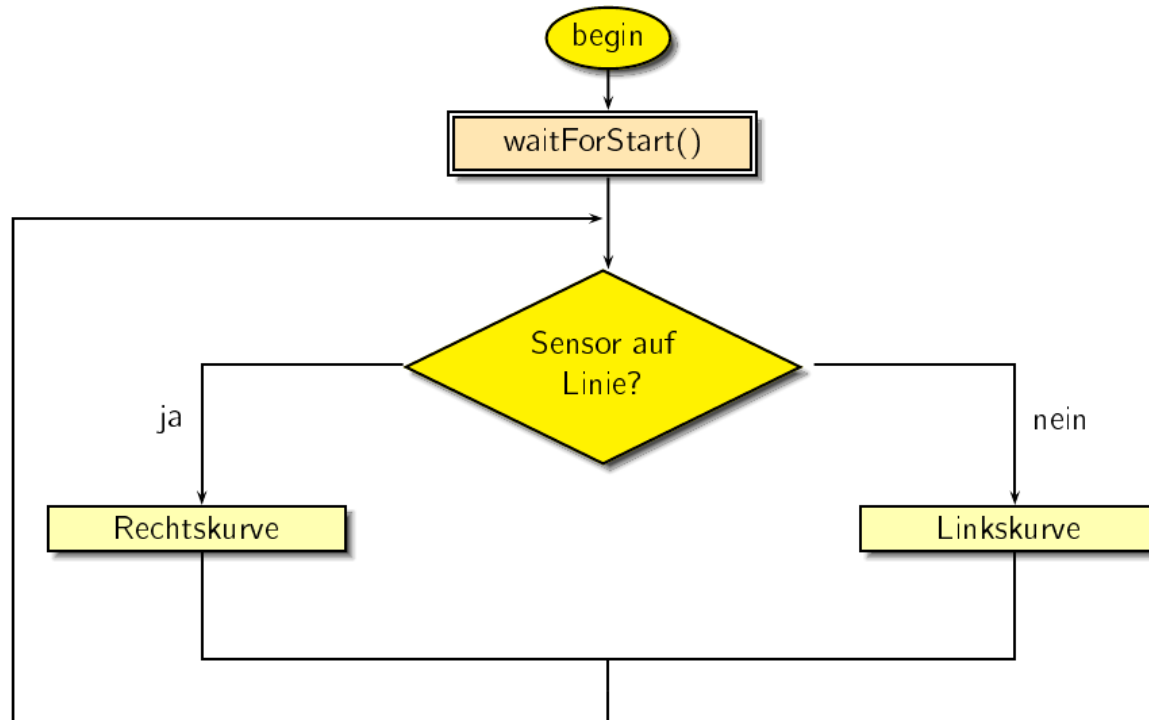
5. Kontrollstrukturen

Allgemein

- Kontrollstrukturen dienen zur Steuerung des Programmablaufs.
- (Bemerkung: C und C++ besitzen die selben Kontrollstrukturen.)
- Ein Programmablauf kann z.B. durch ein Flußdiagramm dargestellt werden.

Flußdiagramme

- Ein Flußdiagramm ist eine grafische Methode zur Beschreibung von Programmabläufen



Grundelemente

- Start
- Stop
- Anweisung
- Fallunterscheidung
- Ein- / Ausgabe

- Pfeile zeigen den Programmablauf an

Anweisungen

- Eine Anweisung ist in jeder Programmiersprache der grundlegende Baustein.
- Beispiele für Anweisungen:
 - `a = 5;`
 - `cout << name;`
 - `cin >> preis;`

Anweisungsblöcke

- Mehrere Anweisungen können durch geschweifte Klammern zu einem Block zusammengefasst werden.

- Beispiel:

```
{  
    a = 5;  
    cout << name;  
    cin >> preis;  
}
```

Fallunterscheidung: if

- Eine Fallunterscheidung dient dazu, Code in Abhängigkeit einer Bedingung auszuführen.
- Formal: `if (<bedingung>)`
`<anweisung>`
- Wenn <bedingung> true ist, wird <anweisung> ausgeführt.
- Beispiel:

```
if (a>b)
    cout << "a ist größer" << endl;
```

Fallunterscheidung: if

- Beispiel mit Anweisungsblock:

```
if (a>b) {  
    cout << "a ist größer" << endl;  
    a = a-b;  
}
```

Fallunterscheidung: if / else

- Wenn in beiden Fällen einer Bedingung (true und false) unterschiedlich weitergearbeitet werden soll, kann `if / else` benutzt werden:

- Formal:

```
if (<bedingung>
    <anweisung1>
else
    <anweisung2>
```

- Beispiel:

```
if (a==b)
    cout << "a gleich b" << endl;
else
    cout << "a ungleich b" << endl;
```

Schachtelung von if / else

- Beispiel:

```
if (a!=b)
    if (a>b)
        cout << "a ist größer" << endl;
    else
        cout << "b ist größer" << endl;
else
    cout << "a gleich b" << endl;
```

Schachtelung von if / else

- „Schöner“: Schachtelung mit Block-Klammern
- Beispiel:

```
if (a!=b) {  
    if (a>b)  
        cout << "a ist größer" << endl;  
    else  
        cout << "b ist größer" << endl;  
}  
else  
    cout << "a gleich b" << endl;
```

Kaskadieren von if / else

- Beispiel:

```
if (a>b)
    cout << "a ist größer" << endl;
else if (b>a)
    cout << "b ist größer" << endl;
else
    cout << "a gleich b" << endl;
```

Switch-Anweisung

- Soll eine Variable auf viele mögliche Werte geprüft werden, ist es manchmal einfacher/eleganter anstatt einer kaskadierten if-Anweisung die switch-Anweisung zu benutzen.
- Beispiel: (if-Kaskade)

```
if (a==1)
    cout << "a ist 1" << endl;
else if (a==2)
    cout << "a ist 2" << endl;
else if (a==3)
    cout << "a ist 3" << endl;
else if (a==4)
    cout << "a ist 4" << endl;
else
    cout << "a ist größer als 4" << endl;
```

Switch-Anweisung

- Eleganter mit switch:

```
switch (a)
{
    case 1:  cout << "a ist 1" << endl;
             break;
    case 2:  cout << "a ist 2" << endl;
             break;
    case 3:  cout << "a ist 3" << endl;
             break;
    case 4:  cout << "a ist 4" << endl;
             break;
    default: cout << "a ist größer 4"
             << endl;
}
```

Switch-Anweisung

- Allgemeine Syntax:

```
switch (<variable>)  
{  
    case <konstante> :    <anweisungsliste>  
                          break;  
    case <konstante> :    <anweisungsliste>  
                          break;  
    case <konstante> :    <anweisungsliste>  
                          break;  
        ...  
    default:              <anweisungsliste>  
}
```

- Der default-Teil darf fehlen.
- Achtung: 'break's nicht vergessen!!

Switch-Anweisung

- Manchmal sind „leere Anweisungslisten“ und sinnvoll.
- Beispiel:

```
switch (c)
{
    case 'a':
    case 'A':      cout << "a" << endl;
                  break;

    case 'b':
    case 'B':      cout << "b" << endl;
                  break;

    case 'c':
    case 'C':      cout << "c" << endl;
                  break;
}
```

Schleifen: while

- Eine Schleife dient dazu, Code in Abhängigkeit einer Bedingung mehrfach auszuführen.
- Formal: `while (<bedingung>)`
`<anweisung>`
- Solange <bedingung> true ist, wird <anweisung> ausgeführt.
- Beispiel:

```
int i = 10;
while (i>0) {
    cout << "i ist " << i << endl;
    i = i-1;
}
```

Schleifen: do / while

- Es kann auch jeweils NACH Ausführung der Anweisung geprüft werden:
- Formal: do
 <anweisung>
 while (<bedingung>)

- Beispiel:

```
int i = 10;  
do {  
    cout << "i ist " << i << endl;  
    i = i-1;  
}  
while (i>0);
```

Endlosschleifen

- Wird als Bedingung in der Schleife

```
while (<bedingung>
    <anweisung>
```

eine Bedingung angeben, die immer true ist, ergibt sich eine Endlosschleife.

- Beispiel:

```
while (1==1) { ... } // immer true
while (true) { ... } // immer true
```

Zählschleifen: for

- Eine Zählschleife (for-Schleife) wird dazu benutzt, Code eine bestimmte Anzahl mal auszuführen.
- Beispiel: $summe = 1+2+3+4+\dots+10$

mit while:

```
int summe=0;
int i=1;      // Schleifenzähler

while (i<=10) {
    summe = summe + i;
    i = i+1;
}
```

Zählschleifen: for

- Beispiel: $summe = 1+2+3+4+\dots+10$

mit for:

```
int summe=0;

for (int i=1; i<=10; i++) {
    summe = summe + i;
}
```

Zählschleifen: for

- Syntax: `for (<start>; <bedingung>; <iteration>)`
`<anweisungsliste>`
- Ablauf:
 - Am Anfang wird einmal `<start>` ausgeführt.
 - Solange `<bedingung>` `true` ist, wird die `<anweisungsliste>` ausgeführt und danach der Iterationsausdruck `<iteration>` ausgeführt.

Zählschleifen: for

- „Grundversion“: n Durchläufe (von 0 bis n-1):

```
for (int i=0; i<n; i++) {  
    // tue irgend etwas  
}
```

Zählschleifen: for

- Bemerkungen:
 - Sollen am Anfang gleich mehrere Variablen initialisiert werden, müssen diese durch Komma getrennt werden:
`for (i=0, j=0; i<5; i++) ...`
 - Natürlich kann auch abwärts gezählt werden:
`for (int i=10; i>=0; i--) ...`
 - Oder jede 2. Zahl:
`for (int i=0; i<20; i=i+2) ...`

break

- Mit der Anweisung `break` lässt sich nicht nur aus einer `switch`-Anweisung springen, sondern auch aus einer `while`- oder `for`-Schleife.
- Beispiel:

```
while (i<1000) {  
    // berechne etwas  
    if (zeit>10) break; // zu lange -> raus  
}
```

continue

- Die Anweisung `continue` beendet die aktuelle Iteration einer `for`- oder `while`-Schleife und startet wieder von oben.
- Beispiel:

```
for (int i=0; i<10; i++) {  
    if (i%2 != 0) continue;  
    cout << i << „ ist gerade“ << endl;  
}
```