

While-Schleife:

Sie dient dazu, eine Abfolge von Anweisungen mehrfach zu wiederholen, solange eine Bedingung erfüllt wird. Die Bedingung wird vor der Ausführung überprüft.

Solange vor Kara kein Baum steht, soll Kara gehen.

```
while (Kara.treeFront())  
{  
    Kara.move();  
}
```

for-Schleife:

Eine for-Schleife ist sinnvoll einzusetzen, wenn eine bestimmte Anzahl von Wiederholungen durchgeführt werden soll, oder wenn eine Variable von einem bestimmten Wert zu einem anderen gezählt werden soll.

```
for (int i=1; i<=9; i++) { ... }
```

Hier macht Kara neunmal ...

Erklärung der verschiedenen Zeichen:

i entspricht einer Variablen, die nur im for-Befehl vorkommt.

$i <= 9$ = solange i kleiner ist als 9, oder = 9

$i++ = i$ wird um 1 erhöht.

While - Schleife:

- Solange die Bedingung in der Klammer erfüllt ist, wird der Code in der while - Schleife ausgeführt
- das ist sinnvoll wenn man bestimmte Sachen bis zu einem Event wiederholen möchte
- while (kara.treefront) { ... }
 - Bedingung
 - Solange Kara vor einem Baum steht, wird dieser Code ausgeführt

Variablen:

- In Variablen kann man Daten speichern. Es gibt lokale und globale Variablen, das hängt davon ab, wo man sie deklariert.
- Man kann sie z.B. als Counter verwenden.
- int counter=0; counter++;

for - Schleife*= Bedingung*

- solange die Condition erfüllt ist, wird der Code in der for - Schleife ausgeführt
- Wenn man einen Vorgang in einer bestimmten Anzahl wiederholen möchte, ist es sinnvoll eine for - Schleife zu benutzen.

- for (int i = 0; i < 5; i++) { ... }

Start Argument

Condition

Argument (wird bei jeder Abfrage der Condition ausgeführt)

hier wird i jeweils um 1 erhöht.

Methoden:

- In Methoden kann man einen Code schreiben, den man durch Schreiben des Methodennamens aufrufen kann
- Wenn man einen Code mehrmals braucht, ist es besser wenn man eine Methode schreibt, weil man die schneller aufrufen kann.
- void zaehlen () { ... }

⋮
public void my program () { zaehlen(); }

Tom, Julius, Alex, Sebastian und Kerstin

Lehrzettel für die JNF-Arbeit

19.11.09

while-Schleife: Solange die Bedingung in den runden Klammern erfüllt ist, werden die Anweisungen in den geschweiften Klammern ausgeführt:

```
while (Bedingung) {  
    Anweisung 1;  
    Anweisung 2;  
}
```

for-Schleife: In den runden Klammern wird eine neue Variable angelegt, die bis zu einem bestimmten Wert hochgezählt wird. Bei jeder Erhöhung der Variable (Variable++) wird der Inhalt der geschweiften Klammern einmal ausgeführt:

```
for (Variable = 1; Variable <= WertVariable 2; Variable++) {  
    Anweisung 1;  
    Anweisung 2;  
}
```

lokale Variablen: lokale Variablen werden in bestimmten Programmteilen (z.B. Methoden) definiert und gelten nur dort.

globale Variablen: Werden innerhalb einer Class ~~definiert~~^{definiert} und gelten ~~für~~ überall.

Methoden

- ohne Rückgabefunktion: Methoden werden mit "void" definiert.

```
void Methoden {  
Anweisung 1;  
Anweisung 2;  
}
```

- mit Rückgabefunktion: Diese Methoden werden mit "int" definiert. Ein Wert einer Variable in wird durch ein "return" in das "int" der Methode kopiert:

```
int Methode() {  
int Variable = 0;  
Anweisung 1;  
Variable++;  
return Variable;  
}
```

Lehrzettel für die JNF-Arbeit

12

19.11.09

- Rekursion: Methoden rufen sich gegenseitig auf oder sich selbst. Die Methoden werden an der Stelle, an der sie die anderen aufrufen, unterbrochen und werden erst ^{zu Ende} ausgeführt, wenn die aufgerufenen fertig sind:

```
void Methode() {  
    Anweisung 1;  
    Anweisung 2;  
    Anweisung 3;  
    if (Bedingung) {  
        Anweisung 4;  
    }  
    else {  
        Methode();  
    }  
    Anweisung 5;  
}
```

- Parameter: Bei einer Methode werden in die runden Klammern ein Wert/eine Variable (mehrere Werte/Variablen) definiert.

- Beim Aufrufen einer Methode kann ein Wert oder eine Variable an dieser Methode übergeben werden:

```
void Methode (int Variable, int Variable 2) {  
  Anweisung 1;  
  Anweisung 2;  
}
```

↓

Im Hauptprogramm:

...
Methode (1, 2);
...

← an dieser Stelle wird
"Methode" aufgerufen
und "Variable" erhält 1 als Inhalt.
"Variable 2" erhält 2

Diese Angaben wie immer ohne Gewähr!!!

© Copyright 2009 by Tom Mergel[®], Felix Albers[®],
Torben Hildebrand[®], Angelo Schwabe[®]
und Jan Florin Kühnel[®]

[®] = All rights reserved