Funktionen in C++

Rekursion

Lindemann

Rekursive Funktionen

Rekursive Funktionen sind solche, die sich selbst aufrufen – also innerhalb der eigenen Definition sich selbst (mit dem eigenen Funktionsnamen) starten.

```
void funk()
{
funk();    // rekursiver Aufruf
}

int main()
{
funk();    // startender Aufruf der Funktion
}
```

Programmabsturz
Stack-Overflow

Rekursive Funktionen

Rekursive Funktionen benötigen daher eine Abbruchbedingung, evtl. über einen Zählparameter

```
1   #include <iostream>
2   using namespace std;
3
4   void funk(int);
5
6   int main()
7   {
8       funk(1);
9   }
10
11  void funk(int zaehler)
12  {
13      if (zaehler <= 10)
14      {
15          cout << "Aufruf Nr.: " << zaehler << endl;
16          funk(zaehler + 1);   // neuer Aufruf mit geändertem Parameter
17      }
18  }
19
```

```
Microsoft Visual Stu
Aufruf Nr.: 1
Aufruf Nr.: 2
Aufruf Nr.: 3
Aufruf Nr.: 4
Aufruf Nr.: 5
Aufruf Nr.: 6
Aufruf Nr.: 7
Aufruf Nr.: 8
Aufruf Nr.: 9
Aufruf Nr.: 10
```

Rekursive Funktionen

Die Rekursion ist ein sehr mächtiges Konzept. So können alle Probleme der Informatik, die iterativ lösbar sind, auch rekursiv gelöst werden (ineffizienter Speicherverbrauch).

```
int Summe(int zahl)
{
 int i, ergebnis = 0;

 for (i = 1; i <= zahl; i++)
  ergebnis = ergebnis + i;

 return ergebnis;
}                    // Iteration
```

```
int SummeR(int zahl)
{
 if (zahl != 0)

  return (zahl + SummeR(zahl - 1));

 else
  return 0;
}                              // Rekursion
```

SummeR(3) 6

```
if (zahl != 0)
   return 3 + SummeR( 3 – 1 )
 else
  . . .
```

3 + 3

```
if (zahl != 0)
   return 2 + SummeR( 2 – 1 )
 else
  . . .
```

2 + 1

```
if (zahl != 0)
   return 1 + SummeR( 1 – 1 )
 else
  . . .
```

1 + 0

```
if (zahl != 0)
   . . .
 else
  return 0;
```

0