Rekursion:

Wenn eine Funktion sich selbst (!) aufruft, spricht man von Rekursion. Bisher wurde von einer Funktion, der main - Funktion, eine andere Funktion aufgerufen. Es ist natürlich möglich, dass diese wiederum noch andere Funktionen aufruft und so weiter.
Es ist aber auch möglich, dass eine Funktion sich selbst aufruft. Das erscheint zunächst nicht sinnvoll kann aber durchaus dienlich sein.
Das übliche Beispiel hierfür ist die Fakultätsfunktion. Sie ist auf natürlichen Zahlen wie folgt definiert:

```
fact(0) = 1
fact(1) = 1
fact(2) = 1*2
fact(3) = 1*2*3
fact(4) = 1*2*3*4
usw.
```

Die Fakultätsfunktion ist in Mathematikbüchern wie folgt definiert:
fact(n) = 1, wenn n = 0 sonst:
fact(n) = n * fact(n-1)

Die **iterative** Lösung könnte wie folgt programmiert werden:

```
int fact(int n)
{
   int result=1;
   do
   {
      result=result*n;
      n=n-1;
   }while(n>0);
   return result;
};
```

Dies lässt sich auch **rekursiv** als C-Programm schreiben:

```
int fact(int n)
{
   if(n==0)
      return 1;
   else
      return n*fact(n-1);
};
```

Beim Programmieren begegnet man der Rekursion ziemlich oft. Rekursive Funktionen brauchen **unbedingt (!)** eine Abbruchbedingung. Im Beispiel mit der Fakultät war die Abbruchbedingung fact(0) = 1.

41. Übung

Compilieren Sie bitte das folgende Programm und analysieren Sie die Funktion des Programmes. Erstellen Sie den zugehörigen Prgrammablaufplan bzw. das zugehörige Struktogramm.

```
#include <iostream>
using namespace std;

void dezdual(int n)
{
     bool bit;
     if (n > 1)
          dezdual (n / 2);
     bit = n % 2;
     cout << bit << " ";
}

int main()
{
     int zahl;
     cout << "* Dezimal - - > Dual  *\n\n"
          << "Geben Sie bitte die dezimale Zahl ein:  ";
     cin >> zahl;
     cout << zahl << "  =  ";
     dezdual (zahl);
     cout << "\n\n";
     getchar();
     getchar();
     return 0;
}
```

Zusatzaufgabe:

Recherchieren Sie bitte, ob die iterative oder die rekursive Lösung das gewünschte Ergebnis schneller berechnet.

Versuchen Sie zu erfahren, warum die eine oder die andere Lösung schneller ist.