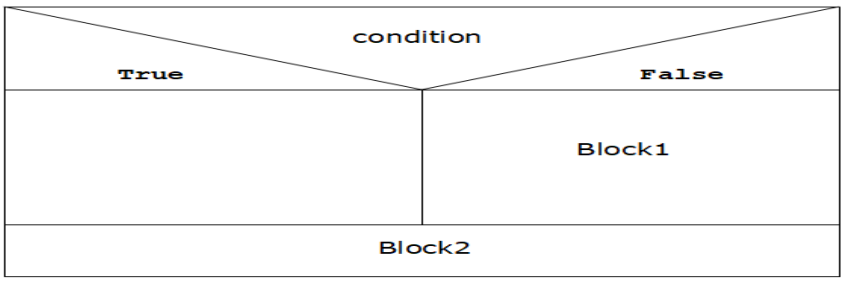
Kursarbeit inf11 **Lösungen** 8.05.2023

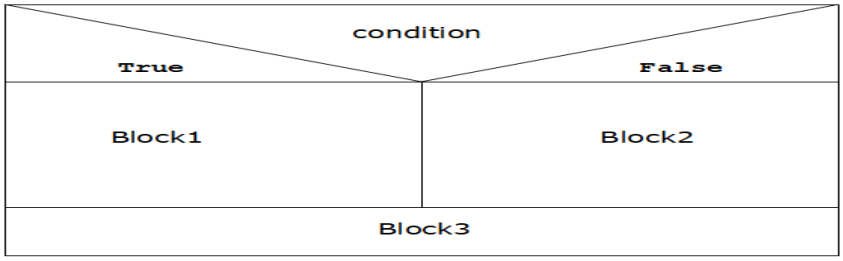
1. In folgenden Struktogrammen bezeichnet **condition** eine Bedingung in Form eines Booleschen Ausdrucks, der die Werte **`True`** oder **`False`** annimmt. Erläutere, worin sich die durch diese Struktogramme gegebenen Algorithmen unterscheiden.

   a)



   condition==True : Nur Block2 wird abgearbeitet.
   condition==False: Block1 und Block2 werden nacheinander abgearbeitet.

   b)



   condition==True : Block1 und Block3 werden nacheinander abgearbeitet.
   condition==False: Block2 und Block3 werden nacheinander abgearbeitet.

2. Ein Python-Programm berechne nach Eingabe einer natürlichen Zahl **n** die Quadrate aller Zahlen 1, ... , n und gebe diese aus.

   a) Der folgende syntaktisch korrekte Python-Quelltext leistet nicht das Verlangte; worin besteht der Fehler, und wie läßt sich dieser beheben?

```
n = int(input('n = ')
i = 1
while i <= n:
    q = i * i
    print (i,'^ 2 =',q)
i = i + 1
```

   Da die Anweisung **`i = i + 1`** *zum Schleifenrumpf gehört, ist wie folgt zu formatieren:*

```
while i <= n:
     q = i * i
     print (i,'^ 2 =',q)
     i = i + 1
```

   b) Welche Ausgabe erfolgt nach Eingabe von n, wenn man den Quellcode nicht abändert?
   Da vor erstmaligem Eintritt in den Schleifenrumpf der Variablen i der Wert 1 zugewiesen wurde und i innerhalb des Schleifenrumpfs nicht geändert wird, ist i<=n immer True, daher erfolgt die wiederholte Ausgabe 1 ^ 2 = 1 *in einer Endlos-Schleife.*

3. Gegeben ist der Algorithmus **POTENZ**, der nach Eingabe einer reellen Zahl **a** und einer natürlichen Zahl **n**, $n \geq 0$, die Potenz $\mathbf{a^n}$ rekursiv berechnet und ausgibt.
   a) Was versteht man unter einer rekursiven Prozedur oder Funktion?

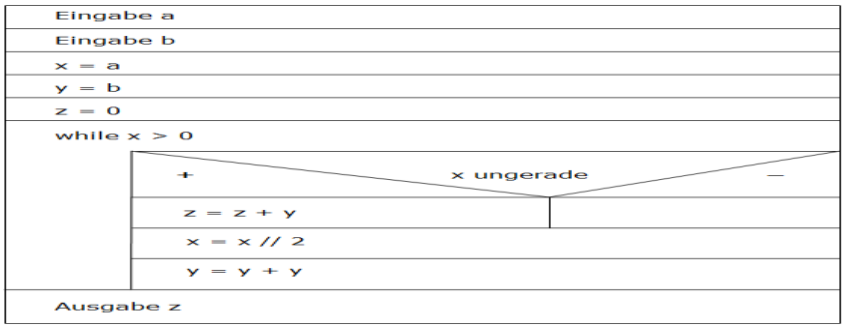
   Eine Prozedur oder eine Funktion, deren Funktionsrumpf mindestens einen Aufruf von sich selbst enthält, heißt rekursiv.

b) Wegen $\mathbf{a^n = a \cdot a^{n-1}}$ läßt sich die Potenzfunktion $\mathbf{f(n) = a^n}$ rekursiv wie folgt definieren:

$\mathbf{f(0) = 1}$ $\quad$ $\mathbf{f(n) = a \cdot f(n-1)}$ falls n > 0

```
a = float(input('a = '))
n = int(input('n = '))
def potenz(x):
      if x == 0: return 1
      else: return a * potenz(x-1)
print (a,'^',n,' =',potenz(n))
```

4. Ein Algorithmus, der die nicht-negativen ganzen Zahlen **a** und **b** als Eingabe verlangt und als Ergebnis die Zahl **z** ausgibt, ist durch folgendes Struktogramm gegeben:



a) Trace-Tabelle für a = 18 und b = 9 als Eingabe:

	a	b	x	y	z	x > 0	x ungerade
vor dem 1. SD	18	9	18	9	0	+	–
vor dem 2. SD	18	9	9	18	0	+	+
vor dem 3. SD	18	9	4	36	18	+	–
vor dem 4. SD	18	9	2	72	18	+	–
vor dem 5. SD	18	9	1	144	18	+	+
nach dem 5. SD	18	9	0	288	162	–	–

Der Algorithmus berechnet vermutlich das Produkt der Zahlen a und b. 18·9=162

b) Quelltext als Python-Programm:

```
a = int(input('a = '))
b = int(input('b = '))
x = a
y = b
z = 0
while x > 0:
    if x % 2 != 0:
        z = z + y
    x = x // 2
    y = y + y
print('z =', z)
```