Übungen zu Prozeduren und Funktionen

1.) Die folgenden vier Programmmodule erfüllen denselben Zweck. Sie berechnen die Oberfläche eines Oktaeders. Geben Sie für jedes Modul den Aufruf an, um die Oberfläche eines Oktaeders mit der Kantenlänge 2 zu berechnen.

```
PROCEDURE Oberflaeche(a:REAL);
BEGIN
 AO := 2*SQR(a)*SQRT(3);
END;
```

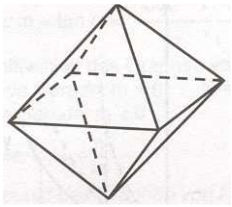
```
PROCEDURE Oberflaeche(a:REAL; VAR AO:REAL);
BEGIN
 AO := 2*SQR(a)*SQRT(3);
END;
```

Aufruf: Oberflaeche(2);
Aufruf: Oberflaeche(2,OF);

```
FUNCTION Oberflaeche(a:REAL):REAL;
BEGIN
 Oberflaeche := 2*SQR(a)*SQRT(3);
END;
```

```
PROCEDURE Oberflaeche;
BEGIN
 AO := 2*SQR(a)*SQRT(3);
END;
```

OF:=Oberflaeche(2);
a:=2; Oberflaeche;



2.) Bewerten Sie Vor- und Nachteile der Programmcodes der vier Module in Übung 1.)!
Modul oben links: Globale Variable AO notwendig, läuft in anderen Programmen nur unter Beachtung von AO,
Modul oben rechts: Ideale Lösung des Problems
Modul unten links: korrekte Lösung, umständlicher als die FUNCTION
Modul unten rechts: schlechteste Variante, nur globale Variablen, läuft in anderen Programmen nur bedingt

Die folgenden Beispiele verdeutlichen noch einmal die Zielstellung des modularen Programmaufbaus. Ein komplexes Problem wird dabei in einfachere Teilprobleme zerlegt und anschließend wieder zusammengesetzt.

*3.) Das Volumen eines Quaders mit den Kantenlängen l=3, b=4, h=5 soll berechnet werden: V = l * b * h*
*Dieses Problem wird mit Hilfe von **FUNCTION** zerlegt in ein Teilproblem Grundfläche und ein Teilproblem Volumen:*

```
FUNCTION Grundflaeche(a, b:REAL):REAL;
BEGIN
 Grundflaeche:=a*b;
END;

FUNCTION Volumen(l, b, h:REAL):REAL;
BEGIN
 Volumen:=Grundflaeche(l , b) * h;
END;                                          Aufruf:      V:=Volumen(3,4,5);
```

4.) Das Volumen eines Zylinders mit dem Radius r=3 und der Höhe h=4 soll berechnet werden: $V = \pi * r^2 * h$
*Dieses Problem wird mit Hilfe von **FUNCTION** zerlegt in ein Teilproblem Grundfläche und ein Teilproblem Volumen:*

```
FUNCTION Grundflaeche(r : REAL):REAL;
BEGIN
 Grundflaeche:=PI * SQR(r);
END;

FUNCTION Volumen(r,h:REAL):REAL;
BEGIN
 Volumen:=Grundflaeche(r) * h;
END;                                          Aufruf:      V:=Volumen(3,4);
```

*5.) Der Flächeninhalt eines beliebigen Dreiecks mit den Kantenlängen a=3, b=5, c=6 soll mit Hilfe der HERON-Formel (siehe TW) berechnet werden. Zerlegen Sie dieses Problem mit Hilfe von **FUNCTION** in mindestens drei Teilprobleme:*

```
FUNCTION Umfang(a,b,c:REAL):REAL;
BEGIN
 Umfang:=a+b+c;
END;

FUNCTION s(a,b,c:REAL):REAL;
BEGIN
 s:=umfang(a,b,c)/2;
END;

FUNCTION Dreieck_Flaeche(a,b,c:REAL):REAL;
BEGIN
 Dreieck_Flaeche:=SQRT(s(a,b,c)*(s(a,b,c)-a)*(s(a,b,c)-b)*(s(a,b,c)-c));
END;
                                              Aufruf:      FDreieck:=Dreieck_Flaeche(3,4,6);
```

*6.) Der Anstieg einer Funktion fx an einer Stelle x=5 soll berechnet werden. Zerlegen Sie dieses Problem mit Hilfe von **FUNCTION** in mindestens zwei Teilprobleme:*

```
FUNCTION fx (x : REAL) : REAL;
{Berechnet den Funktionswert der Funktion y=(x²+3x+5)/(x³-5) an einer Stelle x}
BEGIN
 fx:=(SQR(x) + 3 * x + 5)/(x * SQR(x) - 5);
END;

FUNCTION fxAbleitung (x : REAL) : REAL;
{Berechnet den Anstieg der Funktion y=(x²+3x+5)/(x³-5) an einer Stelle x mit Hilfe
des Differenzenquotienten mit dem Näherungswert h=0.00001}
CONST h = 0.00001;
BEGIN
 fxAbleitung:=(fx(x+h)-fx(x))/h;
END;                                                Aufruf:       Anstieg:=fxAbleitung(5);
```

Funktionen können durchaus auch komplexeren Programmcode umfassen. Ergänzen Sie den Programmcode in den folgenden Funktionen und Prozeduren! Notieren Sie auch immer ein Aufrufbeispiel!

7.)
```
FUNCTION Partialsumme(n : INTEGER) : INTEGER;
{berechnet die Summe der ersten n natürlichen Zahlen}
VAR s,i : INTEGER;
BEGIN
 s:=0;
 FOR i:=1 TO n DO s:=s+i;
 Partialsumme:=s;
END;                                                Aufruf: PS:=Partialsumme(5);
```

8.)
```
FUNCTION nFakultaet(n : INTEGER) : INTEGER;
{berechnet das Produkt der ersten n natürlichen Zahlen}
VAR p,i : INTEGER;
BEGIN
 p:=1;
 FOR i:=1 TO n DO p:=p*i;
 nFakultät:=p;
END;                                                Aufruf: nF:=nFakultät(5);
```

9.)
```
FUNCTION Fibonacci(n : INTEGER):INTEGER;
{Die Funktion gibt das n-te Zahlenfolgenglied der Fibonacci-Folge zurück}
VAR anzahl,m0,m1,i : INTEGER;
BEGIN
 m0:=0; m1:=1; anzahl:=0;
 FOR i:=1 TO n DO
    BEGIN
     anzahl:=m0+m1;
     m0:=m1;
     m1:=anzahl;
    END;
 Fibonacci:=anzahl;
END;                                                Aufruf: F:=Fibonacci(5);
```

10.)
```
FUNCTION Ereignis(Farbe : STRING):STRING;
{Die Prozedur soll je nach Ampelfarbe ‘rot’, ‘gelb’, ‘grün’ den entsprechneden Ereignistext zurückgeben}
BEGIN
 IF Farbe='rot' THEN Ereignis:='Halt'
               ELSE IF Farbe='gelb' THEN Ereignis:='Kreuzung räumen'
                                    ELSE IF Farbe='grün' THEN Ereignis:='Freie Fahrt'
                                                         ELSE Ereignis:='NIL';
END;
                                                    Aufruf: E:=Ereignis(‘rot’);
```

11.) Testen Sie Ihre Funktionen mit einem DELPHI-Projekt FUNKTIONEN_IM_TEST