

☐ Gr. 1, Dr. D. Auer☐ Gr. 2, Dr. G. Kronberger

Gr. 3, Dr. H. Gruber

Name Andreas RoitherAufwand in h 5 h

Punkte \_\_\_\_\_

Kurzzeichen Tutor / Übungsleiter \_\_\_\_\_ / \_\_\_\_\_

**1. Funktionen zur Zeichenkettenverarbeitung****(1 + 2 + 4 Punkte)**

Entwickeln Sie weitere Operationen zur Zeichenkettenbearbeitung (auf dem Standard-Datentyp *STRING*). Verwenden Sie dafür insbesondere die bereits vorhandenen Pascal-Standardfunktionen.

- Implementieren Sie eine Funktion *Reversed*, die als Ergebnis eine Zeichenkette liefert, welche die Zeichen des Eingangsparameters *s* in umgekehrter Reihenfolge enthält.
- Implementieren Sie eine Prozedur *StripBlanks*, die alle Leerzeichen (engl. *blanks*) aus der als Übergangsparameter *s* übergebenen Zeichenkette entfernt.
- Implementieren Sie eine Prozedur *ReplaceAll*, die alle Vorkommen der Zeichenkette *old* (Eingangsparameter) in der Zeichenkette *s* (Übergangsparameter) durch die Zeichenkette *new* (ebenfalls Eingangsparameter) ersetzt und das Ergebnis im Übergangsparameter *s* liefert. Ist *old* nicht in *s* enthalten, muss *s* unverändert bleiben.

**2. Plausibilitätsprüfung von Messwerten****(4 + 3\*1 + 3\*1 Punkte)**

Bei einem Hochofen werden in unregelmäßigen Zeitabständen (alle paar Minuten) Temperaturen gemessen. Dabei wird der Zeitpunkt der Messung (in Stunden und Minuten) sowie die Temperatur (in ° Celsius auf 0,1° genau) ermittelt.

- Entwickeln Sie einen Algorithmus mit Gedächtnis (in Pseudocode), der überprüft, ob ein solcher Temperaturmesswert *plausibel* ist: Ein Temperaturmesswert ist genau dann plausibel, wenn die Minimaltemperatur von 936,8° Celsius nicht unter-, die Maximaltemperatur von 1345,3° Celsius nicht überschritten wird und die Veränderung des Messwerts seit der letzten Messung +/- 11,45° Celsius pro Minute nicht übersteigt.
- Geben Sie drei unterschiedliche Realisierungsmöglichkeiten für Ihren Algorithmus mit Gedächtnis in Pascal an.
- Diskutieren Sie die Vor- und Nachteile dieser drei Realisierungen.

**3. Programmierstil****(2 + 2 + 3 Punkte)**

In der Übung 2, haben Sie sich mit der Berechnung der Quadratwurzel beschäftigt.

Im Licht Ihrer bisherigen (hoffentlich zumindest zum Teil neuen) Erkenntnisse zum Programmieren, zu Pascal und zum Thema Programmierstil:

- Geben Sie Ihre Lösung (also die Lösungsidee und das Pascal-Programm aus der zweiten Übung) völlig unverändert noch einmal an.
- Diskutieren Sie Ihre Lösung möglichst selbstkritisch. Vergleichen Sie Ihre Anmerkungen auch mit jenen, die Sie von der Tutorin / vom Tutor bekommen haben.
- Versuchen Sie nun, es besser zu machen: Geben Sie eine möglichst "gute" Lösungsidee und ein möglichst "gutes" (im Sinne des Programmierstils) Pascal-Programm an.

# Übung 7

## Aufgabe 1

### Lösungsidee

Es werden Operationen zur Zeichenkettenverwaltung erstellt. Reversed liefert mithilfe einer For Schleife einen umgekehrten String zurück. StripBlanks arbeitet mit Call By Reference, und setzt den übergebenen String auf einen neuen String in dem nur Zeichen eingefügt wurden die keine Leerzeichen sind. ReplaceAll arbeitet mit subString und einem result String. An den result String wird ein subString angefügt, der die Zeichen von Position 1 im original String bis zur ersten Position der vorkommenden Zeichenkette -1 (weil sonst das erste Zeichen der Zeichenkette mit kopiert wird), enthält. Danach wird das zu ersetzende Wort angefügt und ein neuer subString erstellt der alle Zeichen, nach der Position + der Länge des alten Wortes, vom Original String enthält. Darauf wird wieder überprüft ob der zu ersetzende String im subString enthalten ist.

```

1 PROGRAM StringOperations;

3 (*Zeichen von einem String werden in einen anderen String kopiert
   Dabei wird von der groessten Stelle runtergezaehlt und das jeweilige
5   Zeichen kopiert*)
FUNCTION Reversed(s : String): String;
7 VAR rev: STRING;
  VAR i : INTEGER;
9 BEGIN
    rev := '';
11   FOR i := Length(s) DOWNT0 1 DO
      BEGIN
13     rev := Concat(rev, s[i]);
      END;
15   Reversed := rev;
  END;

17 (*StripBlanks kopiert nur ein Zeichen in einen anderen String wenn
   dieses Zeichen kein Leerzeichen ist*)
PROCEDURE StripBlanks(VAR s : STRING);
21 VAR i : INTEGER;
  VAR s_new : STRING;
23 BEGIN
    s_new := '';
25   FOR i := 1 TO Length(s) DO
      BEGIN
27     IF s[i] <> ' ' THEN
        BEGIN
29       s_new := Concat(s_new, s[i]);
        END;
31   END;
    s := s_new;
33 END;

35 (*ReplaceAll ersetzt alle vorkommende Strings
   dabei wird immer ein neuer subString erstellt dieser wird
37   wieder überpueft ob old darin vorkommt.*)
PROCEDURE ReplaceAll(old, _new : STRING; VAR s : STRING);

```

```

39 VAR Position, count : INTEGER;
   VAR result, subString : STRING;
41 BEGIN
    Position := Pos(old,s);
43 IF Position <> 0 THEN
    BEGIN
45     result := '';
        count := 0;
47     result := Concat(result,Copy(s,1,Position-1));

49     WHILE Position <> 0 DO
    BEGIN
51
        result := concat(result,_new);
53     subString := Copy(s,(count + Position + length(old)), Length(s));
        count := count + Position + length(old)-1;
55     Position := Pos(old, subString);
        result := Concat(result,Copy(subString,1,Position-1));
57
    END;
59     s := result + subString;
    END;
61 END;

63 VAR s, old, _new : STRING;
   BEGIN
65
    WriteLn(chr(205),chr(205),chr(185), ' StringOperations ',chr(204),chr(205),chr
        (205));
67 WriteLn('Reverse:');
    s := 'Hal lo';
69 WriteLn('Hal lo: ',Reversed(s),#13#10);

71 WriteLn('StripBlanks:');
    StripBlanks(s);
73 WriteLn('Hal lo: ', s,#13#10);

75 WriteLn('ReplaceAll:');
    s := 'Test Hallo das das hierdasenthalten';
77 WriteLn(s);
    ReplaceAll('das','der',s);
79 WriteLn(s);
    ReplaceAll('der','der das',s);
81 WriteLn(s);
END.

```

StringOperations.pas



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>StringOperations.exe
StringOperations
Reverse:
Hal lo: ol laH

StripBlanks:
Hal lo: Hallo

ReplaceAll:
Test Hallo das das hierdasenthalten
Test Hallo der der hierderenthalten
Test Hallo der das der das hierder dasenthalten

C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>
```

Abbildung 1: Testfälle Reverse, StripBlanks, ReplaceAll

## Testfälle

Bei Reverse wird der umgekehrte String ausgegeben. StripBlanks zeigt das Leerzeichen aus einem String entfernt werden. Bei ReplaceAll wird zuerst ein String durch einen komplett anderen ersetzt. Die zweite Ausgabe zeigt das das Ersetzen auch funktioniert wenn der neue String gleich ist mit dem alten oder der alte String in dem neuen enthalten ist.

## Aufgabe 2

### Lösungsidee

Die Plausibilitätsüberprüfung von Messwerten kann auf verschiedene Arten gelöst werden. Die drei hier verwendeten Arten:

- Globale Variablen
- Lokale Variablen
- Unit

Bei Globalen Variablen ist der große Nachteil/Vorteil das sie überall im Programm verwendet werden können. Bei Lokalen Variablen ist der Nachteil das sie nur innerhalb des Hauptprogrammes verwendet werden können. Die Unit verwendet ihre eigenen Variablen und greift auch selbst auf diese zu. Damit kann ohne Variablen Initialisierung im Hauptprogramm trotzdem ein Messwert überprüft werden. Alle drei Arten "merken" die vorherige Zeit und Temperatur und überprüfen damit ob die Temperatur plausibel ist. Zusätzlich müssen bei allen drei Arten die Werte mindestens einmal initialisiert werden sonst funktioniert die Überprüfung nicht.

```

1 FUNCTION isPlausible(h, min : INTEGER; temp : REAL) : BOOLEAN;
2   static lastTime : INTEGER = 0;
3   static lastTemp : Real = 0;
4   VAR diff_time : INTEGER;
5   VAR diff_temp : REAL;
6 BEGIN
7
8   diff_time := (h*60 +min) - lastTime;
9   diff_temp := temp - lastTemp;
10
11  IF (temp < 936,8) OR (temp > 1345,3) OR ((11,45 / 60) * diff_time > abs(
12    diff_temp)) THEN
13    isPlausible := False;
14  ELSE THEN
15    isPlausible := True;
16    lastTime := (h*60 + minute);
17    lastTemp := temp;
18 END;
```

isPlausible\_PseudoCode.pas

```

1 PROGRAM temp;
2
3 USES tempUnit;
4
5 VAR lastTime : INTEGER;
6 VAR lastTemp : REAL;
7
8 FUNCTION isPlausible_Global(h, min : INTEGER; temp : REAL) : BOOLEAN;
9 VAR diff_time : INTEGER;
10 VAR diff_temp : REAL;
11 BEGIN
12   diff_time := ((h * 60) + min) - lastTime;
13   diff_temp := temp - lastTemp;
```

```

15  (*Globale Variablen muessen vorher einmal gesetzt werden*)
    lastTime := (h * 60) + min;
    lastTemp := temp;
17
    isPlausible_Global := True;
19
    IF (temp < 936.8) OR (temp > 1345.3) OR ((11.45 * diff_time) <= abs(diff_temp))
      THEN
21      isPlausible_Global := False;
23 END;

25 FUNCTION isPlausible_Var(h, min : INTEGER; temp : REAL; VAR v_lastTime :
    INTEGER; VAR v_lastTemp : REAL) : BOOLEAN;
    VAR diff_time : INTEGER;
27 VAR diff_temp : REAL;
    BEGIN
29     diff_time := ((h * 60) + min) - v_lastTime;
        diff_temp := temp - v_lastTemp;
31
        v_lastTime := (h * 60) + min;
33     v_lastTemp := temp;

35     isPlausible_Var := True;

37     IF (temp < 936.8) OR (temp > 1345.3) OR ((11.45 * diff_time) <= abs(diff_temp))
        THEN
            isPlausible_Var := False;
39 END;

41 VAR v_lastTime : INTEGER;
    VAR v_lastTemp : REAL;
43 BEGIN
    lastTime := 0;
45     lastTemp := 1000;

47     v_lastTime := 0;
        v_lastTemp := 1000;
49

    WriteLn(chr(205), chr(205), chr(185), ' Temp Plausible ', chr(204), chr(205), chr
        (205));
51

    (*Globale variablen Test*)
53     WriteLn('Global');
        WriteLn('Temp: 1h 0min, Temp: 999, Plausible: ', isPlausible_Global
            (1, 0, 999));
55     WriteLn('Temp: 1h 10min, Temp: 1000, Plausible: ', isPlausible_Global
            (1, 10, 1000));
        WriteLn('Temp: 1h 20min, Temp: 1300, Plausible: ', isPlausible_Global
            (1, 20, 1300));
57     WriteLn('Temp: 2h 0min, Temp: 600, Plausible: ', isPlausible_Global
            (2, 0, 600));
        WriteLn();
59

    (*Lokal variablen Test*)
61     WriteLn('Lokal');
        WriteLn('Temp: 1h 0min, Temp: 999, Plausible: ', isPlausible_Var(1, 0, 999,
            v_lastTime, v_lastTemp));

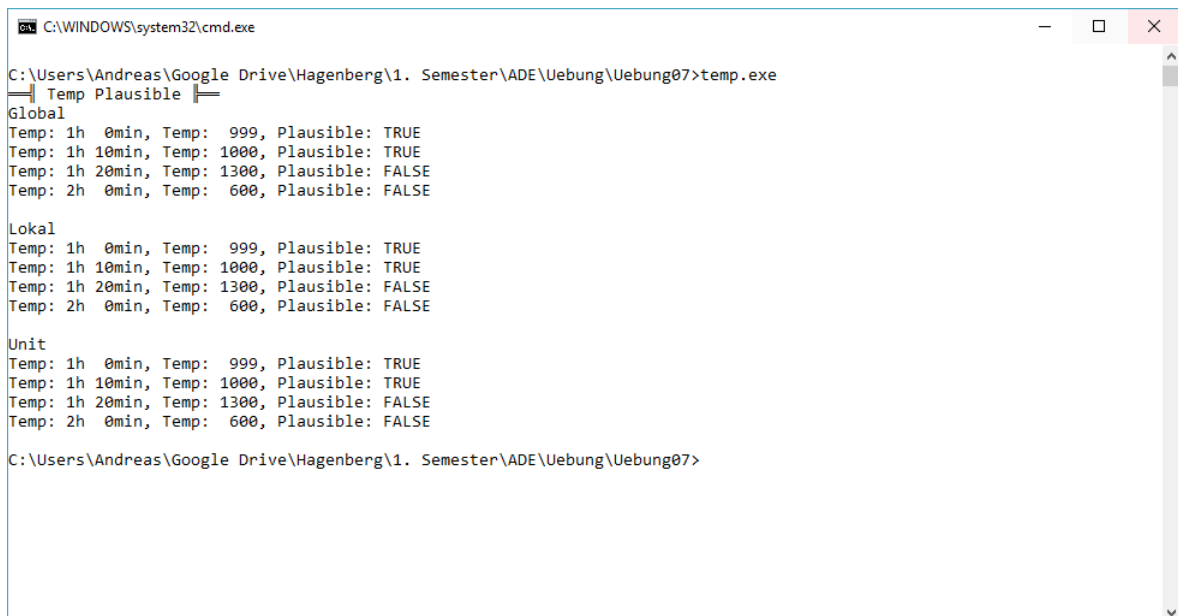
```

```

63 WriteLn( 'Temp: 1h 10min, Temp: 1000, Plausible: ', isPlausible_Var(1,10,1000,
    v_lastTime, v_lastTemp));
WriteLn( 'Temp: 1h 20min, Temp: 1300, Plausible: ', isPlausible_Var(1,20,1300,
    v_lastTime, v_lastTemp));
65 WriteLn( 'Temp: 2h 0min, Temp: 600, Plausible: ', isPlausible_Var(2,0,600,
    v_lastTime, v_lastTemp));
WriteLn();
67
(*UNIT*)
69 WriteLn( 'Unit ');
WriteLn( 'Temp: 1h 0min, Temp: 999, Plausible: ', isPlausible_Unit(1,0,999))
    ;
71 WriteLn( 'Temp: 1h 10min, Temp: 1000, Plausible: ', isPlausible_Unit
    (1,10,1000));
WriteLn( 'Temp: 1h 20min, Temp: 1300, Plausible: ', isPlausible_Unit
    (1,20,1300));
73 WriteLn( 'Temp: 2h 0min, Temp: 600, Plausible: ', isPlausible_Unit(2,0,600))
    ;
75 END.

```

temp.pas



```

C:\WINDOWS\system32\cmd.exe
C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>temp.exe
Temp Plausible
Global
Temp: 1h 0min, Temp: 999, Plausible: TRUE
Temp: 1h 10min, Temp: 1000, Plausible: TRUE
Temp: 1h 20min, Temp: 1300, Plausible: FALSE
Temp: 2h 0min, Temp: 600, Plausible: FALSE
Lokal
Temp: 1h 0min, Temp: 999, Plausible: TRUE
Temp: 1h 10min, Temp: 1000, Plausible: TRUE
Temp: 1h 20min, Temp: 1300, Plausible: FALSE
Temp: 2h 0min, Temp: 600, Plausible: FALSE
Unit
Temp: 1h 0min, Temp: 999, Plausible: TRUE
Temp: 1h 10min, Temp: 1000, Plausible: TRUE
Temp: 1h 20min, Temp: 1300, Plausible: FALSE
Temp: 2h 0min, Temp: 600, Plausible: FALSE
C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>

```

Abbildung 2: Testfälle Temperatur Messung

## Testfälle

Die Testfälle zeigen die Messwertüberprüfung bei allen drei Arten.

## Aufgabe 3

### Lösungsidee Übung 2

Bei der Quadratwurzelberechnung soll mithilfe einer Reellen Zahl und einer Fehlerschranke die Quadratische Wurzel der Reellen Zahl berechnet werden. Dazu muss die Richtigkeit der Eingabe überprüft werden und entsprechende Fehlermeldungen ausgegeben werden. Nach der Überprüfung der Eingabe soll mithilfe der Newtonsche Iterationsformel eine Näherung für  $y \approx \sqrt{x}$  berechnet werden. Mithilfe der Formel  $y_1 = \frac{1}{2}(y_0 + \frac{x}{y_0})$  und einer Repeat Until Schleife die nach 50 Iterationen abbricht wird eine Näherung für die Eingabe berechnet.

```

1 program quadratwurzel_old;
var r_digit, approx_error, y0, y1: Real;
3 var count: Integer;
begin
5   Write('— Quadratwurzel —', #13#10, 'Reelle Zahl eingeben: ');
   Read(r_digit);
7   Write(#13);
   Write('Fehlerschranke eingeben: ');
9   Read(approx_error);
   Write('_____', #13#10);

11
   if r_digit <= 0 then
13     Write('Fehler: erste Eingabe kleiner als 0', #13#10); (*#10#13 sind
       Steuerbefehle für die Konsole*)

15
   if approx_error = 0 then
       Write('Fehler: Fehlerschranke ist 0', #13#10)

17
   else
19     begin
       y1 := 1; (*Startwert*)
21     count := 0;
       repeat
23       y0 := y1;
       y1 := (y0 + (r_digit / y0))/2;
25       count := count + 1;
       (*Abbruch der Schleife nach 50 Iterationen*)
27     until (abs(y1 - y0) <= approx_error) or (count = 50);

29     if count = 50 then
       Write('Fehler: Nach 50 Iterationen keine Konvergenz gefunden')
31     else begin
       Write('Naeherung: ', y1, ' nach ', count, ' Iterationen');
33     end;
   end;
35   Write(#13#10);
end.

```

quadratwurzel\_old.pas



## Anmerkungen

Für die Lösungsidee ist nicht notwendig wie diese Formel aussieht solange sie im Programm verwendet wird. Der Programmstil selbst zeichnet sich durch einen nicht eingehaltenen Standard aus, wie etwa das klein schreiben der Schlüsselwörter. Die Näherung selbst kann in eine eigene Funktion ausgelagert werden, damit kann die Funktion überall einfach verwendet werden, falls Änderungen an der Näherung gemacht werden muss somit nicht überall der Programm Code verändert werden. Die Repeat Until Schleife kann durch eine While Schleife ersetzt werden.

## Lösungsidee Neu

Mithilfe einer Prozedur und Call By Reference wird mit einer Näherungsformel die Quadratwurzel einer Zahl ausgerechnet. Wenn nach 50 Iterationen keine Konvergenz gefunden wurde wird abgebrochen. Zusätzlich werden die Eingaben auf Fehler überprüft und entsprechende Fehlermeldungen ausgegeben.

```

1 PROGRAM quadratwurzel_new;

3 PROCEDURE Naeherung(VAR y0,y1, r_digit : REAL);
  BEGIN
5     y0 := y1;
     y1 := (y0 + (r_digit / y0))/2;
7 END;

9 VAR r_digit , approx_error , y0, y1: REAL;
  VAR count: INTEGER;
11 BEGIN
    WriteLn(chr(205),chr(205),chr(185), ' Quadratwurzel_new ',chr(204),chr(205),
      chr(205));
13 WriteLn('Reelle Zahl eingeben: ');
    Read(r_digit);
15 Write(#13);
    Write('Fehlerschranke eingeben: ');
17 Read(approx_error);
    Write('_____', #13#10);


19 IF r_digit <= 0 THEN
21 Write('Fehler: erste Eingabe kleiner als 0', #13#10);
    IF approx_error = 0 THEN
23 Write('Fehler: Fehlerschranke ist 0', #13#10)
    ELSE BEGIN
25 y1 := 1;
     y0 := 1;
27 count := 0;
     WHILE (count < 50) AND (abs(y1 - y0) >= approx_error) DO
29 BEGIN
        Naeherung(y0,y1,r_digit);
31 count := count + 1;
     END;
33 END;

35 IF count = 50 THEN
    Write('Fehler: Nach 50 Iterationen keine Konvergenz gefunden')

```

```
37 ELSE
    Write( 'Naehierung: ', y1, ' nach ', count, ' Iterationen ');
39
    Write(#13#10);
41 END.
```

quadratwurzel\_new.pas



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>quadratwurzel_new.exe
Quadratwurzel_new
Reelle Zahl eingeben:
64
Fehlerschranke eingeben: 0.0000000001
-----
Naehierung: 8.000000000000000E+000 nach 8 Iterationen
C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>quadratwurzel_new.exe
Quadratwurzel_new
Reelle Zahl eingeben:
123456712345612345672134567123456123456
Fehlerschranke eingeben: 0.00000000000001
-----
Fehler: Nach 50 Iterationen keine Konvergenz gefunden
C:\Users\Andreas\Google Drive\Hagenberg\1. Semester\ADE\Uebung\Uebung07>
```

Abbildung 3: Testfälle Quadratwurzel

## Testfälle

Die Testfälle zeigen die Ausgabe mit der überarbeiteten Version von Quadratwurzel.