


Übung 2

Aufgabe 1

```
1 program spannweite;  
2 var min, max, val, spanwidth : Integer;  
3 begin  
4   Write('Spannweiten Berechnung', #13#10, 'Zahl eingeben: ');  
5   Read(val);  
6   Write(#13);  
7   if val > 0 then  
8     begin  
9       min := val;  
10      max := val;  
11      repeat  
12        Write('Zahl eingeben: ');  
13        Read(val);  
14        Write(#13);  
15        if val > 0 then  
16          begin  
17            if val >= max then  
18              max := val;  
19            else if val < min then  
20              min := val;  
21            end  
22          until val <= 0;  
23  
24      if max = min then  
25        Write('Spannweite: 0', #13#10)  
26      else  
27        begin  
28          spanwidth := max - min;  
29          Write('Spannweite: ', spanwidth, #13#10);  
30        end  
31      end  
32    else  
33      Write('Spannweite: 0', #13#10);  
34  end.
```



```
C:\WINDOWS\system32\cmd.exe

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>spannweite.exe
Spannweiten Berechnung
Zahl eingeben: 1
Zahl eingeben: 5
Zahl eingeben: 8
Zahl eingeben: 2
Zahl eingeben: 0
Spannweite: 7

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>spannweite.exe
Spannweiten Berechnung
Zahl eingeben: 8
Zahl eingeben: 3
Zahl eingeben: 0
Spannweite: 5

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>spannweite.exe
Spannweiten Berechnung
Zahl eingeben: 0
Spannweite: 0

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>spannweite.exe
Spannweiten Berechnung
Zahl eingeben: 3
Zahl eingeben: 4
Zahl eingeben: -1
Spannweite: 1

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>
```

Abbildung 1: Testfälle Spannweitenberechnung

Testfälle

Bei der Spannweitenberechnung wurden 4 mögliche Testfälle ausprobiert. Die Reihenfolge der Zahlen, negative Zahlen und Nulleingaben wurden berücksichtigt und getestet.

Aufgabe 2

```
program spannweite;
2 var a, b ,c , lowest , mid, highest: Integer;
begin
4   Write('— Zahlen Sortierung —', #13#10 , 'Zahl eingeben: ');
   Read(a);
6   Write(#13);
   lowest := a;
8   mid := 0;
   highest := 0;
10  Write('Zahl eingeben: ');
   Read(b);
12  Write(#13);
   Write('Zahl eingeben: ');
14  Read(c);

16  if b < lowest then
begin
18     mid := lowest;
     lowest := b;
20 end
else
22     mid := b;

24  if c >= mid then
     highest := c
26  else if (c < mid) And (c >= lowest) then
begin
28     highest := mid;
     mid := c;
30 end
else if c < lowest then
32 begin
     highest := mid;
     mid := lowest;
     lowest := c;
34 end;
36

38 Write('Zahlen aufsteigend sortiert: ' , lowest , ' ' , mid , ' ' , highest);
   Write(#13#10);
40 end.
```



```
C:\WINDOWS\system32\cmd.exe

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>sortieren.exe
-- Zahlen Sortierung --
Zahl eingeben: 1
Zahl eingeben: 3
Zahl eingeben: 6
Zahlen aufsteigend sortiert: 1 3 6

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>sortieren.exe
-- Zahlen Sortierung --
Zahl eingeben: -1
Zahl eingeben: 3
Zahl eingeben: 5
Zahlen aufsteigend sortiert: -1 3 5

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>sortieren.exe
-- Zahlen Sortierung --
Zahl eingeben: 6
Zahl eingeben: 5
Zahl eingeben: 1
Zahlen aufsteigend sortiert: 1 5 6

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>sortieren.exe
-- Zahlen Sortierung --
Zahl eingeben: 0
Zahl eingeben: 1
Zahl eingeben: 5
Zahlen aufsteigend sortiert: 0 1 5

C:\Users\andir\Google Drive\Hagenberg\Software Engineering\1. Semester\ADE\Übung\Übung 2>
```

Abbildung 2: Testfälle Sortieralgorithmus

Testfälle

Für den Sortieralgorithmus wurde die Reihenfolge der Zahlen, negative Zahlen, und Nulleingabe getestet.

Aufgabe 3

Lösungsidee

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.

```

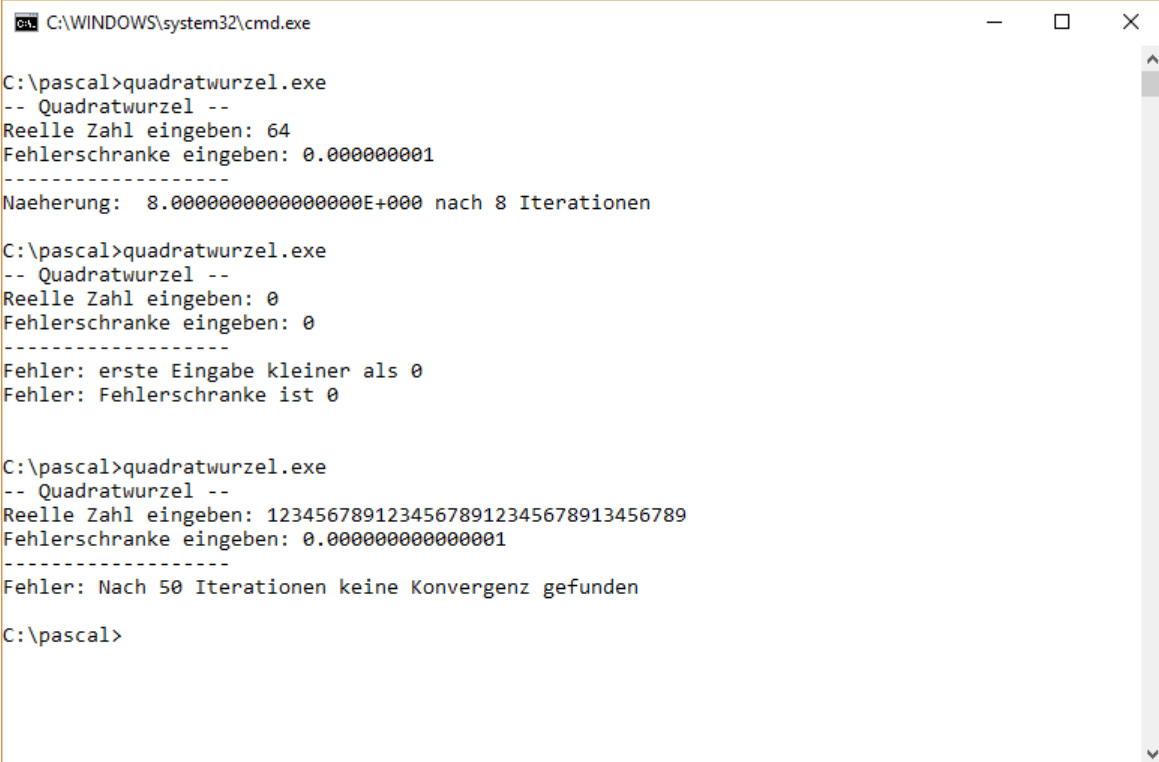
program quadratwurzel;
2 var r_digit, approx_error, y0, y1: Real;
  var count: Integer;
4 begin
  Write('— Quadratwurzel —', #13#10, 'Reelle Zahl eingeben: ');
6  Read(r_digit);
  Write(#13);
8  Write('Fehlerschranke eingeben: ');
  Read(approx_error);
10 Write('_____', #13#10);

12 if r_digit <= 0 then
  Write('Fehler: erste Eingabe kleiner als 0', #13#10); (*#10#13 sind
    Steuerbefehle für die Konsole*)
14
16 if approx_error = 0 then
  Write('Fehler: Fehlerschranke ist 0', #13#10)

18 else
  begin
20    y1 := 1; (*Startwert*)
    count := 0;
22    repeat
      y0 := y1;
24      y1 := (y0 + (r_digit / y0))/2;
      count := count + 1;
26      (*Abbruch der Schleife nach 50 Iterationen*)
    until (abs(y1 - y0) <= approx_error) or (count = 50);

28    if count = 50 then
      Write('Fehler: Nach 50 Iterationen keine Konvergenz gefunden')
30    else begin
      Write('Naeherung: ', y1, ' nach ', count, ' Iterationen');
32      end;
34    end;
    Write(#13#10);
36 end.

```



```
C:\WINDOWS\system32\cmd.exe

C:\pascal>quadratwurzel.exe
-- Quadratwurzel --
Reelle Zahl eingeben: 64
Fehlerschranke eingeben: 0.000000001
-----
Naeherung: 8.000000000000000E+000 nach 8 Iterationen

C:\pascal>quadratwurzel.exe
-- Quadratwurzel --
Reelle Zahl eingeben: 0
Fehlerschranke eingeben: 0
-----
Fehler: erste Eingabe kleiner als 0
Fehler: Fehlerschranke ist 0

C:\pascal>quadratwurzel.exe
-- Quadratwurzel --
Reelle Zahl eingeben: 12345678912345678912345678913456789
Fehlerschranke eingeben: 0.000000000000001
-----
Fehler: Nach 50 Iterationen keine Konvergenz gefunden

C:\pascal>
```

Abbildung 3: Testfälle Quadratwurzelberechnung

Testfälle

Bei der Quadratwurzelberechnung wurde auf verschiedene Fehlerquellen getestet. Der erste Testfall dient zur generellen Überprüfung der Berechnung, der zweite Testfall dient zur Überprüfung der Eingabe. Der dritte Testfall soll zeigen das nach 50 Iterationen abgebrochen wird.