

☒ Gr. 1, S. Schöberl, MScName Elias Leonhardsberger Aufwand in h 6☐ Gr. 2, DI (FH) G. Horn-Völlenkne, MSc

Punkte _____ Tutor*in / Übungsleiter*in ____ / ____

1. Skalieren von einfachen Computergrafiken

(24 Punkte)

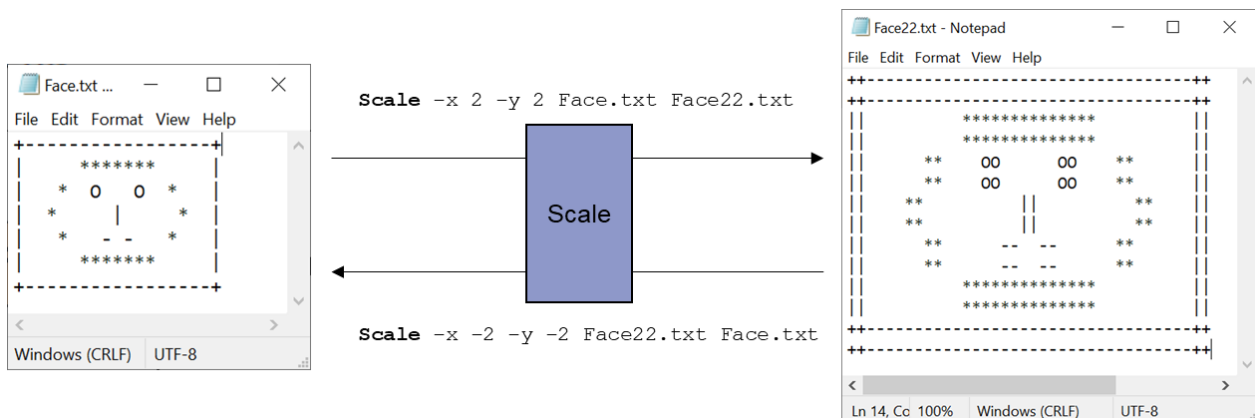
In Textdateien kann man nicht nur Texte, sondern auch sehr einfache Computergrafiken, wie zum Beispiel ASCII-Art (siehe <https://www.asciart.eu/>) speichern. Entwickeln Sie ein Programm *Scale*, das den Inhalt einer Textdatei einliest, in horizontaler (x) und vertikaler (y) Richtung skaliert und das Ergebnis in einer weiteren Datei speichert. Der Skalierungsfaktor für die x - und y -Richtung soll über die Kommandozeile angegeben werden: Positive Werte im Bereich 2 bis 9 bewirken eine Vergrößerung. Negative Werte im Bereich -2 bis -9 bewirken eine Verkleinerung, wobei der Wert -2 die Größe auf die Hälfte reduziert, -3 auf ein Drittel usw. Ihr Programm muss folgende Aufrufmöglichkeiten von der Kommandozeile bieten (die Metasymbole [...] kennzeichnen optionale Parameter):

Scale [-x sx] [-y sy] inFile outFile

Bedeutung der Parameter:

-x sx	der Skalierungsfaktor sx in x-Richtung (Standardwert 1),
-y sy	der Skalierungsfaktor sy in y-Richtung (Standardwert 1),
inFile	Name der Eingabedatei und
outFile	Name der Ausgabedatei.

Beispiel:



Bei der Verkleinerung hängt das Ergebnis davon ab, welches Zeichen in x -Richtung und welche Zeile in y -Richtung aus dem Original ausgewählt und in den verkleinerten Bereich übertragen wird. Soll z.B. der Inhalt der Datei *Face22.txt* um den Faktor $1/3$ in x -Richtung verkleinert werden, könnte man aus den ersten drei Zeichen der ersten Zeile ++- das Zeichen + oder - auswählen. Für diese Aufgabe genügt eine einfache Strategie, die beispielsweise immer das erste Zeichen auswählt. Dies gilt sinngemäß auch für die Verkleinerung in y -Richtung.

Hinweise:

1. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
2. Dokumentieren und kommentieren Sie Ihre Algorithmen.
3. Bei Programmen: Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Programm funktioniert, und dass es auch in Fehlersituation entsprechend reagiert.

ADF2/PRO2 UE02

Elias Leonhardsberger

24. April 2024, Hagenberg

Inhaltsverzeichnis

1	Skalieren von einfachen Computergrafiken	4
1.1	Lösungsidee	4
1.2	Source Code	5
1.2.1	ULineBuffer.pas	5
1.2.2	Scale.pas	7
1.3	Tests	12
1.3.1	Testskript	12
1.3.2	baseAsciiArt.txt	17
1.3.3	testfile2.txt	17
1.3.4	testfile3.txt	18
1.3.5	testfile4.txt	18
1.3.6	testfile5.txt	18
1.3.7	testfile6.txt	18
1.3.8	testfile7.txt	19
1.3.9	testfile8.txt	19
1.3.10	testfile9.txt	20
1.3.11	reallyBigFile.txt	20
1.4	Testergebnisse	21
1.4.1	Ausgabe des Testskripts	21
1.4.2	scaledAsciiArt.txt	26
1.4.3	resultfile2.txt	26
1.4.4	resultfile2+x.txt	27
1.4.5	resultfile2+y.txt	27
1.4.6	resultfile2+xy.txt	28
1.4.7	resultfile2+yx.txt	28
1.4.8	resultfile2-x.txt	29
1.4.9	resultfile2-y.txt	29
1.4.10	resultfile2-xy.txt	30
1.4.11	resultfile2-yx.txt	30
1.4.12	resultfile2+x-y.txt	30
1.4.13	resultfile2-x+y.txt	30
1.4.14	resultfile2+y-x.txt	31
1.4.15	resultfile2-y+x.txt	32

1.4.16	resultfile-9.txt	32
1.4.17	resultfile-8.txt	32
1.4.18	resultfile-7.txt	32
1.4.19	resultfile-6.txt	32
1.4.20	resultfile-5.txt	32
1.4.21	resultfile-4.txt	33
1.4.22	resultfile-3.txt	33
1.4.23	resultfile3.txt	33
1.4.24	resultfile4.txt	33
1.4.25	resultfile5.txt	34
1.4.26	resultfile6.txt	35
1.4.27	resultfile7.txt	37
1.4.28	resultfile8.txt	39
1.4.29	resultfile9.txt	42
1.4.30	reallyBigFile.txt	45

1 Skalieren von einfachen Computergrafiken

1.1 Lösungsidee

Ein LineBuffer wird erstellt, der alle hinzugefügten *STRINGS* mithilfe einer verketteten Liste hintereinander hängt. Zuerst wäre eine Trennung der hinzugefügten *STRINGS*, um den Platz des Buffers optimal auszunutzen, vorgesehen. Da aber durch das Read() eines Files 255 Zeichen lange *STRINGS* gelesen werden, mit der einzigen Ausnahmen am Ende einer Zeile, und da die Implementierung noch Fehler beinhaltete, wurde diese Behandlung wieder gestrichen.

Beim Lesen der Datei wird jede Zeile zuerst in den LineBuffer eingelesen und dann beim Schreiben skaliert, indem entweder jedes x-te Zeichen geschrieben wird oder jedes Zeichen x-mal geschrieben wird.

Die Skalierung in Y Richtung funktioniert ähnlich, nur statt Zeichen werden ganze Zeilen ans gelassen oder vervielfacht.

Eingabe Parameter werden durchlaufen und mit mehreren Verzweigungen überprüft um die Optionalität und Reihenfolge der einzelnen Parametern zu gewährleisten. Falls das Programm falsch aufgerufen wird, wird eine Help Ausgabe angezeigt um den User über die richtige Verwendung zu informieren.

Die Tests sind in einem shell Script geschrieben und mit folgendem Befehl ausgeführt.

```
./ TestScale.sh &> ./ TestScaleOutput.txt
```

Durch die 9-fache Skalierung entstehen leider sehr große und unübersichtliche Ergebnisse, welche dennoch in diesem Dokument sind, um alle Ergebnisse zu zeigen.

1.2 Souce Code

1.2.1 ULineBuffer.pas

```
1
2  UNIT ULineBuffer;
3
4  INTERFACE
5
6  TYPE
7      LineBufferPrt = ^LineBufferNode;
8      LineBufferNode = RECORD
9          buffer : STRING;
10         next : LineBufferPrt;
11     END;
12     LineBuffer = ^LineBufferNode;
13
14  PROCEDURE InitLineBuffer(VAR lb: LineBuffer);
15  PROCEDURE DisposeLineBuffer(VAR lb: LineBuffer);
16  PROCEDURE ClearLineBuffer(VAR lb: LineBuffer);
17  PROCEDURE AppendToLineBuffer(VAR lb: LineBuffer; line: STRING);
18  PROCEDURE WriteLineBuffer(VAR outFile: TEXT; VAR lb: LineBuffer; scale:
    ↪  INTEGER);
19
20  IMPLEMENTATION
21
22  PROCEDURE InitLineBuffer(VAR lb: LineBuffer);
23  BEGIN (* InitLineBuffer *)
24      NEW(lb);
25      lb^.buffer := '';
26      lb^.next := NIL;
27  END; (* InitLineBuffer *)
28
29  PROCEDURE DisposeLineBuffer(VAR lb: LineBuffer);
30  VAR
31      next: LineBufferPrt;
32  BEGIN (* DisposeLineBuffer *)
33      WHILE (lb <> NIL) DO
34          BEGIN (* WHILE *)
35              next := lb^.next;
36              DISPOSE(lb);
37              lb := next;
38          END; (* WHILE *)
39  END; (* DisposeLineBuffer *)
40
41  PROCEDURE ClearLineBuffer(VAR lb: LineBuffer);
42  VAR
43      temp, next: LineBufferPrt;
```

```

44 BEGIN (* ClearLineBuffer *)
45     temp := lb^.next;
46
47     WHILE (temp <> NIL) DO
48         BEGIN (* WHILE *)
49             next := temp^.next;
50             DISPOSE(temp);
51             temp := next;
52         END; (* WHILE *)
53
54     lb^.buffer := '';
55     lb^.next := NIL;
56 END; (* ClearLineBuffer *)
57
58 PROCEDURE AppendToLineBuffer(VAR lb: LineBuffer; line: STRING);
59 VAR
60     prev, newNode: LineBufferPrt;
61 BEGIN (* AppendToLineBuffer *)
62     prev := lb;
63
64     WHILE (prev^.next <> NIL) DO
65         BEGIN (* WHILE *)
66             prev := prev^.next;
67         END; (* WHILE *)
68
69     IF (prev^.buffer = '') THEN
70         BEGIN (* IF *)
71             prev^.buffer := line;
72         END (* IF *)
73     ELSE
74         BEGIN (* ELSE *)
75             NEW(newNode);
76             newNode^.next := NIL;
77             newNode^.buffer := line;
78             prev^.next := newNode;
79         END; (* ELSE *)
80 END;
81
82 PROCEDURE WriteScaledLine(VAR outFile: TEXT; line: STRING; scale:
83     ↪ INTEGER; VAR countModScale: INTEGER);
84 VAR
85     i, j: INTEGER;
86 BEGIN (* WriteScaledLine *)
87     FOR i := 1 TO Length(line) DO
88         BEGIN (* FOR *)
89             IF (scale > 0) THEN

```

```

90         FOR j := 1 TO scale DO
91             BEGIN (* FOR *)
92                 write(outFile, line[i]);
93             END; (* FOR *)
94         END (* IF *)
95     ELSE
96         IF ((countModScale MOD scale) = 0) THEN
97             BEGIN (* ELSE IF *)
98                 write(outFile, line[i]);
99             END; (* ELSE IF *)
100
101         countModScale := (countModScale + 1) MOD scale;
102     END; (* FOR *)
103 END; (* WriteScaledLine *)
104
105 PROCEDURE WriteLineBuffer(VAR outFile: TEXT; VAR lb: LineBuffer; scale:
    ↪ INTEGER);
106 VAR
107     temp: LineBufferPrt;
108     countModScale: INTEGER;
109 BEGIN (* WriteLineBuffer *)
110     temp := lb;
111     countModScale := 0;
112
113     WHILE (temp <> NIL) DO
114         BEGIN (* WHILE *)
115             IF (scale = 1) THEN
116                 BEGIN (* IF *)
117                     write(outFile, temp^.buffer);
118                 END (* IF *)
119             ELSE
120                 BEGIN (* ELSE *)
121                     WriteScaledLine(outFile, temp^.buffer, scale, countModScale)
122                 END; (* ELSE *)
123
124                 temp := temp^.next;
125             END; (* WHILE *)
126
127     writeln(outFile);
128 END; (* WriteLineBuffer *)
129
130 END.

```

1.2.2 Scale.pas

```

1 PROGRAM Scale;
2
3 USES

```

```

4  ULineBuffer;
5
6  PROCEDURE ScaleFile(VAR inFile, outFile: TEXT; scaleX, scaleY: INTEGER);
7  VAR
8      lb: LineBuffer;
9      s: STRING;
10     countModScale, i: INTEGER;
11 BEGIN (* ScaleFile *)
12     countModScale := 0;
13     InitLineBuffer(lb);
14
15     WHILE NOT EOF(inFile) DO
16         BEGIN (* WHILE *)
17             WHILE NOT EOLN(inFile) DO
18                 BEGIN (* WHILE *)
19                     Read(inFile, s);
20                     AppendToLineBuffer(lb, s);
21                 END;
22
23                 IF (scaleY > 0) THEN
24                     BEGIN (* IF *)
25                         FOR i := 1 TO scaleY DO
26                             BEGIN (* FOR *)
27                                 WriteLineBuffer(outFile, lb, scaleX);
28                             END; (* FOR *)
29                         END (* IF *)
30                     ELSE
31                         IF (countModScale MOD scaleY = 0) THEN
32                             BEGIN (* ELSE IF *)
33                                 WriteLineBuffer(outFile, lb, scaleX);
34                             END; (* ELSE IF *)
35
36                             ClearLineBuffer(lb);
37                             ReadLn(inFile);
38                             countModScale := (countModScale + 1) MOD scaleY;
39                         END; (* WHILE *)
40
41                     DisposeLineBuffer(lb);
42                 END; (* ScaleFile *)
43
44 PROCEDURE ShowHelp;
45 BEGIN (* ShowHelp *)
46     WriteLn('Usage: Scale [OPTION] inFile outFile');
47     WriteLn('    inFile: input file. ');
48     WriteLn('    outFile: output file. ');
49     WriteLn('    -x sx: scale in the x direction. Default 1. Allowed values:
        ↪ -9 to -2, 2 to 9. ');

```



```

50   WriteLn('    -y sy: scale in the y direction. Default 1. Allowed values:
      ↪ -9 to -2, 2 to 9.');
```

```

51   WriteLn('    --help: display this help and exit.');
```

```

52 END; (* ShowHelp *)
```

```

53
```

```

54 PROCEDURE GetParameters(VAR scaleX, scaleY: INTEGER; VAR inFileName,
      ↪ outFileName: STRING);
```

```

55 VAR
```

```

56   xSet, ySet, inFileSet, outFileSet: BOOLEAN;
```

```

57   i, errorCode: INTEGER;
```

```

58 BEGIN (* GetParameters *)
```

```

59   scaleX := 1;
```

```

60   scaleY := 1;
```

```

61   xSet := FALSE;
```

```

62   ySet := FALSE;
```

```

63   inFileSet := FALSE;
```

```

64   outFileSet := FALSE;
```

```

65   i := 1;
```

```

66
```

```

67 IF (ParamCount < 2) OR (ParamStr(1) = '--help') THEN
```

```

68   BEGIN (* IF *)
```

```

69     ShowHelp();
```

```

70     HALT(1);
```

```

71   END; (* IF *)
```

```

72
```

```

73 WHILE (i <= ParamCount) DO
```

```

74   BEGIN (* WHILE *)
```

```

75     IF ((NOT xSet) AND ((i + 1) < ParamCount) AND (ParamStr(i) = '-x'))
```

```

76       ↪ THEN
```

```

77       BEGIN (* IF *)
```

```

78         Val(ParamStr(i + 1), scaleX, errorCode);
```

```

79
```

```

80         IF ((errorCode <> 0) OR (scaleX < -9) OR (scaleX > 9) OR
```

```

81           ↪ (scaleX = 0)) THEN
```

```

82           BEGIN (* IF *)
```

```

83             WriteLn(StdErr, 'Invalid scale value for x direction.');
```

```

84             ShowHelp();
```

```

85             HALT(1);
```

```

86           END; (* IF *)
```

```

87
```

```

88           xSet := TRUE;
```

```

89           i := i + 2;
```

```

90         END (* IF *)
```

```

91       ELSE
```

```

92         IF ((NOT ySet) AND ((i + 1) < ParamCount) AND (ParamStr(i) =
```

```

93           ↪ '-y')) THEN
```

```

94           BEGIN (* ELSE IF *)
```

```

92         Val(ParamStr(i + 1), scaleY, errorCode);
93
94         IF ((errorCode <> 0) OR (scaleY < -9) OR (scaleY > 9) OR
95         ↪ (scaleY = 0)) THEN
96             BEGIN (* IF *)
97                 WriteLn(StdErr, 'Invalid scale value for y direction.');
```

ShowHelp();

```

98                 HALT(1);
99             END; (* IF *)
100
101         ySet := TRUE;
102         i := i + 2;
103     END (* ELSE IF *)
104 ELSE
105     IF ((NOT inFileSet) AND (ParamStr(i) <> '-x') AND (ParamStr(i) <>
106     ↪ '-y')) THEN
107         BEGIN (* ELSE IF *)
108             inFileName := ParamStr(i);
109             inFileSet := TRUE;
110             i := i + 1;
111         END (* ELSE IF *)
112     ELSE
113         IF ((NOT outFileSet) AND (ParamStr(i) <> '-x') AND (ParamStr(i)
114         ↪ <> '-y')) THEN
115             BEGIN (* ELSE IF *)
116                 outFileName := ParamStr(i);
117                 outFileSet := TRUE;
118                 i := i + 1;
119             END (* ELSE IF *)
120         ELSE
121             BEGIN (* ELSE *)
122                 ShowHelp();
123                 HALT(1);
124             END; (* ELSE *)
125     END;
126
127     IF ((NOT inFileSet) OR (NOT outFileSet)) THEN
128         BEGIN (* IF *)
129             ShowHelp();
130             HALT(1);
131         END; (* IF *)
132     END; (* GetParameters *)
133
134 VAR
135     inFile, outFile: TEXT;
136     errorCode: WORD;
137     scaleX, scaleY: INTEGER;
```

```

136     inFileName, outFileName: STRING;
137 BEGIN (* Scale *)
138     GetParameters(scaleX, scaleY, inFileName, outFileName);
139
140     Assign(inFile, inFileName);
141     {$I-}
142     Reset(inFile);
143     {$I+}
144     errorCode := IOResult;
145
146     IF (errorCode <> 0) THEN
147         BEGIN (* IF *)
148             writeln(StdErr, 'Error while opening input file. ');
149             writeln(StdErr, 'Error code: ', errorCode);
150             HALT(1);
151         END; (* IF *)
152
153     Assign(outFile, outFileName);
154     {$I-}
155     Rewrite(outFile);
156     {$I+}
157     errorCode := IOResult;
158
159     IF (errorCode <> 0) THEN
160         BEGIN (* IF *)
161             writeln(StdErr, 'Error while opening output file. ');
162             writeln(StdErr, 'Error code: ', errorCode);
163             HALT(1);
164         END; (* IF *)
165
166     ScaleFile(inFile, outFile, scaleX, scaleY);
167
168     WriteLn('File scaled successfully. ');
169
170     Close(inFile);
171     Close(outFile);
172 END. (* Scale *)

```

1.3 Tests

1.3.1 Testskript

```
1  echo "No parameters"
2  bin/Scale
3
4  echo ""
5  echo "No optional parameters"
6  bin/Scale TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
7
8  echo ""
9  echo "X optional parameter"
10 bin/Scale -x 1 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
11
12 echo ""
13 echo "Y optional parameter"
14 bin/Scale -y 1 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
15
16 echo ""
17 echo "X and Y optional parameters"
18 bin/Scale -x 1 -y 1 TestFiles/baseAsciiArt.txt
   ↪ ResultFiles/scaledAsciiArt.txt
19
20 echo ""
21 echo "Y and X optional parameters"
22 bin/Scale -y 1 -x 1 TestFiles/baseAsciiArt.txt
   ↪ ResultFiles/scaledAsciiArt.txt
23
24 echo ""
25 echo "X and Y optional parameters with different signs"
26 bin/Scale -y +1 -x -1 TestFiles/baseAsciiArt.txt
   ↪ ResultFiles/scaledAsciiArt.txt
27
28 echo ""
29 echo "Wrong optional parameter"
30 bin/Scale -z 1 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
31
32 echo ""
33 echo "Too many optional parameters"
34 bin/Scale -x 1 -y 1 -x 1 TestFiles/baseAsciiArt.txt
   ↪ ResultFiles/scaledAsciiArt.txt
35
36 echo ""
37 echo "No file parameters"
38 bin/Scale -x 1 -y 1
39
40 echo ""
```

```

41  echo "One file parameter"
42  bin/Scale -x 1 -y 1 ResultFiles/baseAsciiArt.txt
43
44  echo ""
45  echo "Invalid input file"
46  bin/Scale -x 1 -y 1 invalidFile ResultFiles/scaledAsciiArt.txt
47
48  echo ""
49  echo "X = 0"
50  bin/Scale -x 0 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
51
52  echo ""
53  echo "X = 10"
54  bin/Scale -x 10 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
55
56  echo ""
57  echo "X = -10"
58  bin/Scale -x -10 TestFiles/baseAsciiArt.txt
   ↪ ResultFiles/scaledAsciiArt.txt
59
60  echo ""
61  echo "Y = 0"
62  bin/Scale -y 0 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
63
64  echo ""
65  echo "Y = 10"
66  bin/Scale -y 10 TestFiles/baseAsciiArt.txt ResultFiles/scaledAsciiArt.txt
67
68  echo ""
69  echo "Y = -10"
70  bin/Scale -y -10 TestFiles/baseAsciiArt.txt
   ↪ ResultFiles/scaledAsciiArt.txt
71
72  echo ""
73  echo "X = 1, Y = 1"
74  bin/Scale TestFiles/testfile2.txt ResultFiles/resultfile2.txt
75
76  echo ""
77  echo "X = 2, Y = 1"
78  bin/Scale -x 2 TestFiles/testfile2.txt ResultFiles/resultfile2+x.txt
79
80  echo ""
81  echo "X = 1, Y = 2"
82  bin/Scale -y 2 TestFiles/testfile2.txt ResultFiles/resultfile2+y.txt
83
84  echo ""
85  echo "X = 2, Y = 2"

```

```

86 bin/Scale -x 2 -y 2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2+xy.txt
87
88 echo ""
89 echo "Y = 2, X = 2"
90 bin/Scale -y 2 -x 2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2+yx.txt
91
92 echo ""
93 echo "X = -2, Y = 1"
94 bin/Scale -x -2 TestFiles/testfile2.txt ResultFiles/resultfile2-x.txt
95
96 echo ""
97 echo "X = 1, Y = -2"
98 bin/Scale -y -2 TestFiles/testfile2.txt ResultFiles/resultfile2-y.txt
99
100 echo ""
101 echo "X = -2, Y = -2"
102 bin/Scale -x -2 -y -2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2-xy.txt
103
104 echo ""
105 echo "Y = -2, X = -2"
106 bin/Scale -y -2 -x -2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2-yx.txt
107
108 echo ""
109 echo "X = 2, Y = -2"
110 bin/Scale -x 2 -y -2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2+x-y.txt
111
112 echo ""
113 echo "X = -2, Y = 2"
114 bin/Scale -x -2 -y 2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2-x+y.txt
115
116 echo ""
117 echo "Y = 2, X = -2"
118 bin/Scale -y 2 -x -2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2+y-x.txt
119
120 echo ""
121 echo "Y = -2, X = 2"
122 bin/Scale -y -2 -x 2 TestFiles/testfile2.txt
   ↪ ResultFiles/resultfile2-y+x.txt
123
124 echo ""

```

```

125 echo "Scale by -9"
126 bin/Scale -x -9 -y -9 TestFiles/testfile9.txt
    ↳ ResultFiles/resultfile-9.txt
127
128 echo ""
129 echo "Scale by -8"
130 bin/Scale -x -8 -y -8 TestFiles/testfile8.txt
    ↳ ResultFiles/resultfile-8.txt
131
132 echo ""
133 echo "Scale by -7"
134 bin/Scale -x -7 -y -7 TestFiles/testfile7.txt
    ↳ ResultFiles/resultfile-7.txt
135
136 echo ""
137 echo "Scale by -6"
138 bin/Scale -x -6 -y -6 TestFiles/testfile6.txt
    ↳ ResultFiles/resultfile-6.txt
139
140 echo ""
141 echo "Scale by -5"
142 bin/Scale -x -5 -y -5 TestFiles/testfile5.txt
    ↳ ResultFiles/resultfile-5.txt
143
144 echo ""
145 echo "Scale by -4"
146 bin/Scale -x -4 -y -4 TestFiles/testfile4.txt
    ↳ ResultFiles/resultfile-4.txt
147
148 echo ""
149 echo "Scale by -3"
150 bin/Scale -x -3 -y -3 TestFiles/testfile3.txt
    ↳ ResultFiles/resultfile-3.txt
151
152 echo ""
153 echo "Scale by 3"
154 bin/Scale -x 3 -y 3 TestFiles/testfile3.txt ResultFiles/resultfile3.txt
155
156 echo ""
157 echo "Scale by 4"
158 bin/Scale -x 4 -y 4 TestFiles/testfile4.txt ResultFiles/resultfile4.txt
159
160 echo ""
161 echo "Scale by 5"
162 bin/Scale -x 5 -y 5 TestFiles/testfile5.txt ResultFiles/resultfile5.txt
163
164 echo ""

```

```
165 echo "Scale by 6"
166 bin/Scale -x 6 -y 6 TestFiles/testfile6.txt ResultFiles/resultfile6.txt
167
168 echo ""
169 echo "Scale by 7"
170 bin/Scale -x 7 -y 7 TestFiles/testfile7.txt ResultFiles/resultfile7.txt
171
172 echo ""
173 echo "Scale by 8"
174 bin/Scale -x 8 -y 8 TestFiles/testfile8.txt ResultFiles/resultfile8.txt
175
176 echo ""
177 echo "Scale by 9"
178 bin/Scale -x 9 -y 9 TestFiles/testfile9.txt ResultFiles/resultfile9.txt
179
180 echo ""
181 echo "BIG FILE"
182 bin/Scale -x 9 -y 9 TestFiles/reallyBigFile.txt
    ↪ ResultFiles/reallyBigFile.txt
```


1.3.2 baseAsciiArt.txt

Listing 1: baseAsciiArt.txt

The drawing is a complex, abstract line drawing. It features a large, stylized letter 'L' on the right side, formed by a vertical line and a horizontal line. A long horizontal line extends from the left side of the 'L' towards the left edge of the drawing. This horizontal line is composed of several segments, some of which are dashed or dotted. On the left side of the drawing, there are several smaller, interconnected shapes, some of which are also formed by dashed or dotted lines. There are several small text elements scattered throughout the drawing, including the letter 'L' in two different locations, the letter 'Y' in the center, and various symbols like '(>)' and 'L'. The drawing is framed by a thick, textured border at the top and bottom.

1.3.3 testfile2.txt

Listing 2: testfile2.txt

```

X.X.X.X.X.X.X.
. . . . .
X.X.X.X.X.X.X.
. . . . .
X.X.X.X.X.X.X.
. . . . .

```

```

X.X.X.X.X.X.X.
.....
X.X.X.X.X.X.X.
.....
X.X.X.X.X.X.X.
.....
X.X.X.X.X.X.X.
.....
1.3.4 testfile3.txt

```

Listing 3: testfile3.txt

```

X..X..
.....
.....
X..X..
.....
.....
1.3.5 testfile4.txt

```

Listing 4: testfile4.txt

```

X...X...
.....
.....
.....
X...X...
.....
.....
.....
1.3.6 testfile5.txt

```

Listing 5: testfile5.txt

```

X....X....
.....
.....
.....
.....
X....X....
.....
.....
.....
.....
1.3.7 testfile6.txt

```

Listing 6: testfile6.txt

```

X.....X.....

```

```

.....
.....
.....
.....
.....
X.....X.....
.....
.....
.....
.....
.....

```

1.3.8 testfile7.txt

Listing 7: testfile7.txt

```

X.....X.....
.....
.....
.....
.....
.....
.....
X.....X.....
.....
.....
.....
.....
.....
.....

```

1.3.9 testfile8.txt

Listing 8: testfile8.txt

```

X.....X.....
.....
.....
.....
.....
.....
.....
X.....X.....
.....
.....
.....
.....
.....
.....

```

.....
1.3.10 testfile9.txt

Listing 9: testfile9.txt

```
X.....X.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
X.....X.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....
```

1.3.11 reallyBigFile.txt

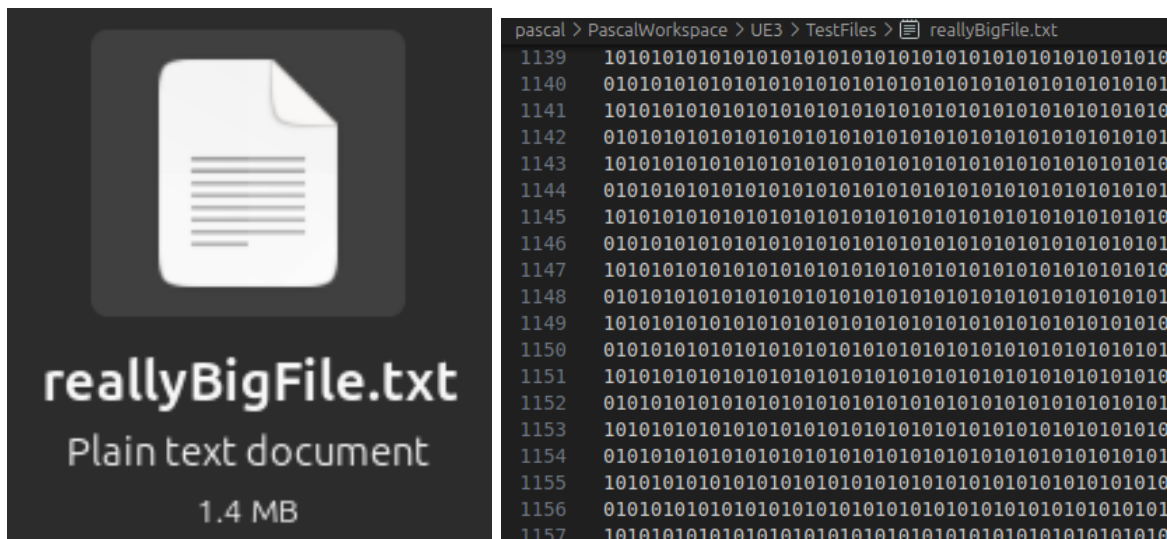


Abbildung 1: Ausschnitt der Testdatei "reallyBigFile.txt"

1.4 Testergebnisse

1.4.1 Ausgabe des Testskripts

Listing 10: TestScaleOutput.txt

No parameters

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

No optional parameters

File scaled successfully.

X optional parameter

File scaled successfully.

Y optional parameter

File scaled successfully.

X and Y optional parameters

File scaled successfully.

Y and X optional parameters

File scaled successfully.

X and Y optional parameters with different signs

File scaled successfully.

Wrong optional parameter

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Too many optional parameters

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

No file parameters

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

One file parameter

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid input file

Error while opening input file.

Error code: 2

X = 0

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid scale value for x direction.

X = 10

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid scale value for x direction.

X = -10

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid scale value for x direction.

Y = 0

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid scale value for y direction.

Y = 10

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid scale value for y direction.

Y = -10

Usage: Scale [OPTION] inFile outFile

inFile: input file.

outFile: output file.

-x sx: scale in the x direction. Default 1. Allowed values: -9 to -2, 2

-y sy: scale in the y direction. Default 1. Allowed values: -9 to -2, 2

—help: display this help and exit.

Invalid scale value for y direction.

X = 1, Y = 1

File scaled successfully.

X = 2, Y = 1

File scaled successfully.

X = 1, Y = 2

File scaled successfully.

X = 2, Y = 2

File scaled successfully.

Y = 2, X = 2

File scaled successfully.

X = -2, Y = 1

File scaled successfully.

X = 1, Y = -2

File scaled successfully.

$X = -2, Y = -2$
File scaled successfully.

$Y = -2, X = -2$
File scaled successfully.

$X = 2, Y = -2$
File scaled successfully.

$X = -2, Y = 2$
File scaled successfully.

$Y = 2, X = -2$
File scaled successfully.

$Y = -2, X = 2$
File scaled successfully.

Scale by -9
File scaled successfully.

Scale by -8
File scaled successfully.

Scale by -7
File scaled successfully.

Scale by -6
File scaled successfully.

Scale by -5
File scaled successfully.

Scale by -4
File scaled successfully.

Scale by -3
File scaled successfully.

Scale by 3
File scaled successfully.

Scale by 4
File scaled successfully.

Scale by 5

File scaled successfully .

Scale by 6

File scaled successfully .

Scale by 7

File scaled successfully .

Scale by 8

File scaled successfully .

Scale by 9

File scaled successfully .

BIG FILE

File scaled successfully .

1.4.2 scaledAsciiArt.txt

Listing 11: scaledAsciiArt.txt

1.4.3 resultfile2.txt

Listing 12: resultfile2.txt

```

X.X.X.X.X.X.X.
. . . . .
X.X.X.X.X.X.X.
. . . . .
X.X.X.X.X.X.X.
. . . . .

```

```

X.X.X.X.X.X.X.
.....
X.X.X.X.X.X.X.
.....
X.X.X.X.X.X.X.
.....
X.X.X.X.X.X.X.
.....

```

1.4.4 resultfile2+x.txt

Listing 13: resultfile2+x.txt

```

XX..XX..XX..XX..XX..XX..XX..
.....
XX..XX..XX..XX..XX..XX..XX..
.....
XX..XX..XX..XX..XX..XX..XX..
.....
XX..XX..XX..XX..XX..XX..XX..
.....
XX..XX..XX..XX..XX..XX..XX..
.....
XX..XX..XX..XX..XX..XX..XX..
.....
XX..XX..XX..XX..XX..XX..XX..
.....

```

1.4.5 resultfile2+y.txt

Listing 14: resultfile2+y.txt

```

X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
.....
.....
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
.....
.....
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
.....
.....
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
.....
.....
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.

```

```

.....
.....
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
.....
.....
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
.....
.....

```

1.4.6 resultfile2+xy.txt

Listing 15: resultfile2+xy.txt

```

XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....

```

1.4.7 resultfile2+yx.txt

Listing 16: resultfile2+yx.txt

```

XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..

```

```

.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
.....
.....

```

1.4.8 resultfile2-x.txt

Listing 17: resultfile2-x.txt

```

XXXXXXXX
.....
XXXXXXXX
.....
XXXXXXXX
.....
XXXXXXXX
.....
XXXXXXXX
.....
XXXXXXXX
.....

```

1.4.9 resultfile2-y.txt

Listing 18: resultfile2-y.txt

```

X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.

```

1.4.10 resultfile2-xy.txt

Listing 19: resultfile2-xy.txt

```

XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX

```

1.4.11 resultfile2-yx.txt

Listing 20: resultfile2-yx.txt

```

XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX

```

1.4.12 resultfile2+x-y.txt

Listing 21: resultfile2+x-y.txt

```

XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..

```

1.4.13 resultfile2-x+y.txt

Listing 22: resultfile2-x+y.txt

```

XXXXXXXX
XXXXXXXX
.....
.....

```

```

XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .

```

1.4.14 resultfile2+y-x.txt

Listing 23: resultfile2+y-x.txt

```

XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .
XXXXXXXX
XXXXXXXX
. . . . .
. . . . .

```

```
XXXXXXX
XXXXXXX
. . . . .
. . . . .
XXXXXXX
XXXXXXX
. . . . .
. . . . .
```

1.4.15 resultfile2-y+x.txt

Listing 24: resultfile2-y+x.txt

```
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
XX..XX..XX..XX..XX..XX..XX..
```

1.4.16 resultfile-9.txt

Listing 25: resultfile-9.txt

```
XX
XX
```

1.4.17 resultfile-8.txt

Listing 26: resultfile-8.txt

```
XX
XX
```

1.4.18 resultfile-7.txt

Listing 27: resultfile-7.txt

```
XX
XX
```

1.4.19 resultfile-6.txt

Listing 28: resultfile-6.txt

```
XX
XX
```

1.4.20 resultfile-5.txt

Listing 29: resultfile-5.txt

```
XX
XX
```


1.4.21 resultfile-4.txt

Listing 30: resultfile-4.txt

XX
XX

1.4.22 resultfile-3.txt

Listing 31: resultfile-3.txt

XX
XX

1.4.23 resultfile3.txt

Listing 32: resultfile3.txt

XXX.....XXX.....
XXX.....XXX.....
XXX.....XXX.....
.....
.....
.....
.....
.....
.....
XXX.....XXX.....
XXX.....XXX.....
XXX.....XXX.....
.....
.....
.....
.....
.....
.....

1.4.24 resultfile4.txt

Listing 33: resultfile4.txt

XXXX.....XXXX.....
XXXX.....XXXX.....
XXXX.....XXXX.....
XXXX.....XXXX.....
.....
.....
.....
.....
.....
.....
.....

XXXX	XXXX
XXXX	XXXX
XXXX	XXXX
XXXX	XXXX

1.4.25 resultfile5.txt

Listing 34: resultfile5.txt

[illegible]

[illegible]

1.4.26 resultfile6.txt

Listing 35: resultfile6.txt

[illegible]

[illegible]

[illegible]

[illegible]

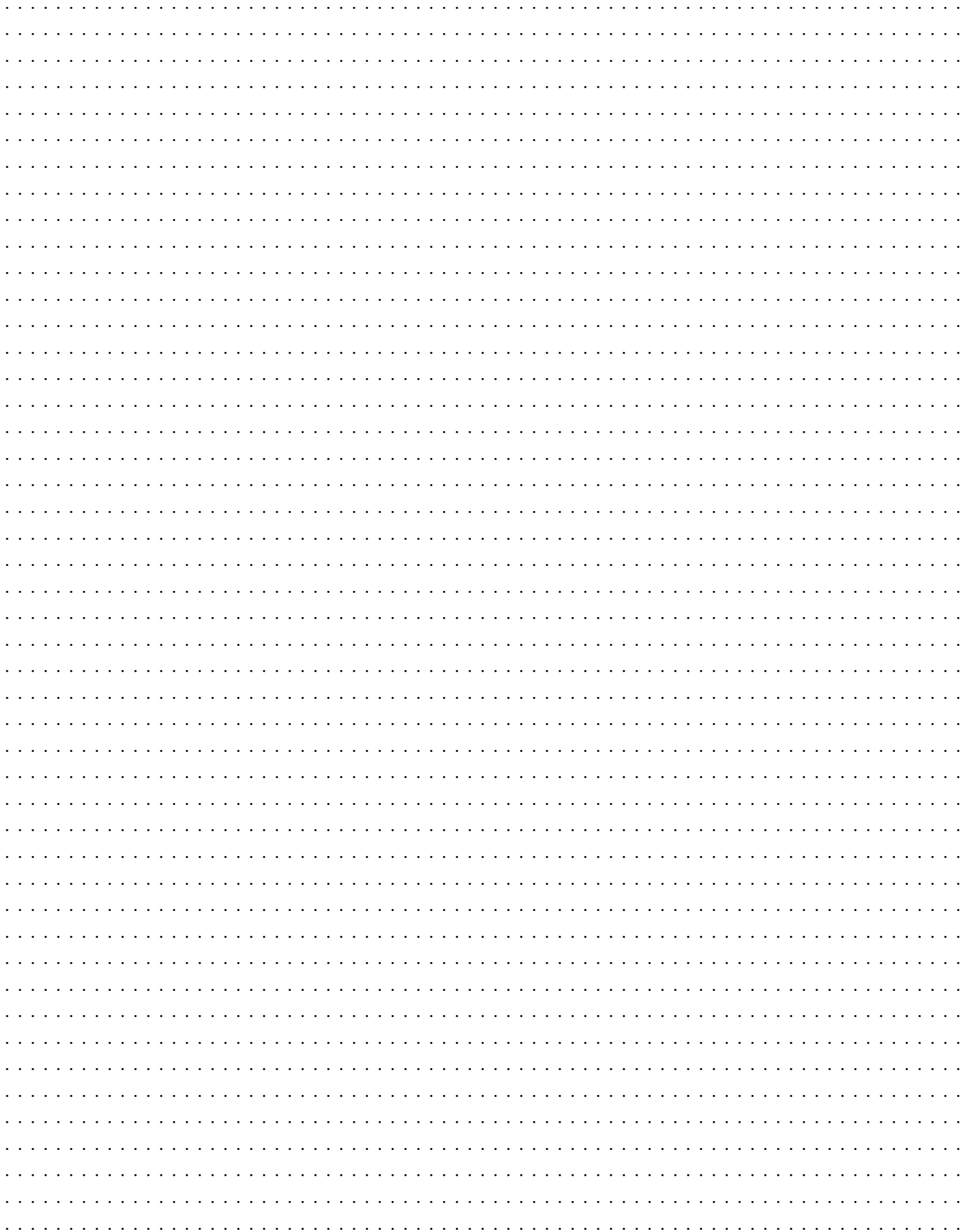
1.4.28 resultfile8.txt

Listing 37: resultfile8.txt

[illegible]

This image shows a full page of dot grid paper. The background is white, and it is covered with a regular pattern of small, dark grey dots. The dots are arranged in straight horizontal and vertical lines, creating a grid of small squares across the entire page. There are no margins, text, or other markings present.

[illegible]



1.4.30 reallyBigFile.txt



Abbildung 2: Ausschnitt der Ergebnisdatei "reallyBigFile.txt"