

☐ Gr. 1, Dr. G. KronbergerName Andreas Roither Aufwand in h 6 h☐ Gr. 2, Dr. H. Gruber

Gr. 3, Dr. D. Auer

Punkte Kurzzeichen Tutor / Übungsleiter / **1. (De-)Kompression von Dateien****(10 Punkte)**

Die *Lauf längencodierung* (engl. *run length encoding*, kurz *RLE*) ist eine einfache Kompressionstechnik für Dateien, bei der jede Zeichenfolge, die aus mehr als zwei gleichen Zeichen besteht, durch das erste Zeichen und die Länge der Folge codiert wird.

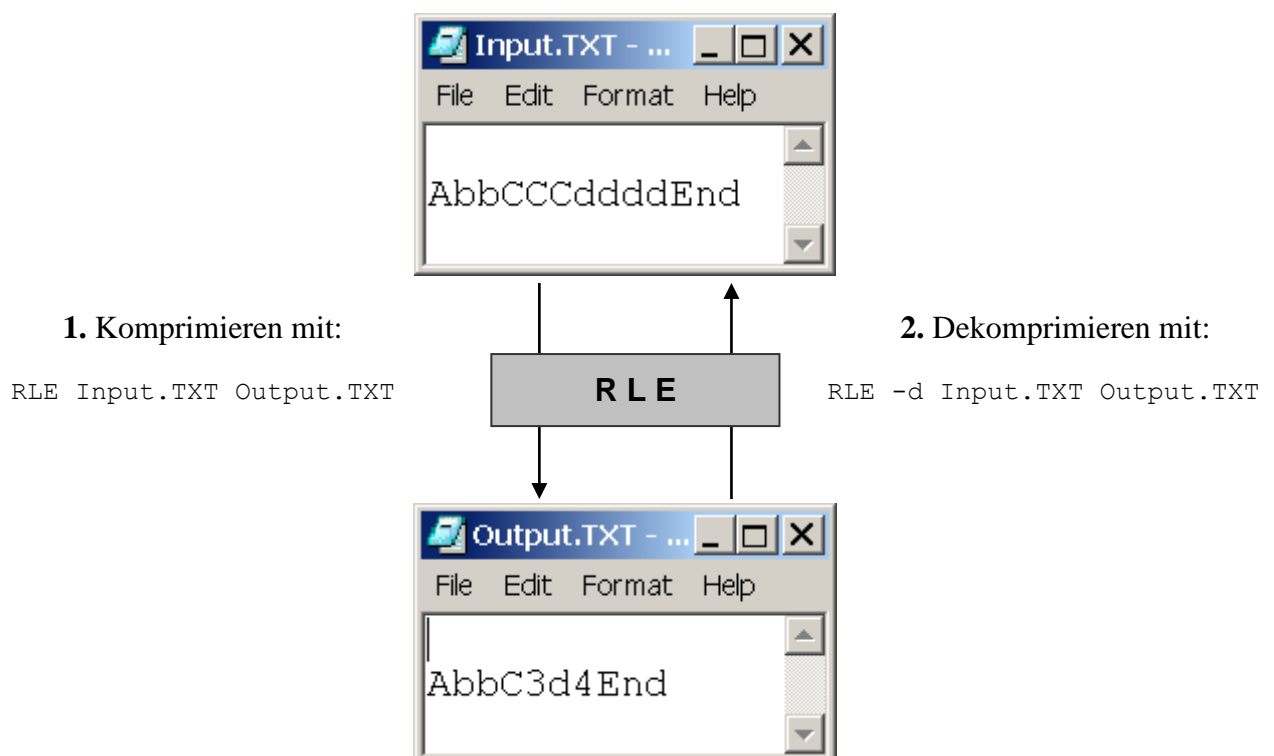
Implementieren Sie eine einfache Variante dieses Verfahrens in Form eines Filterprogramms *RLE*, das Textdateien, die nur Groß- und Kleinbuchstaben, Satz- und das Leerzeichen (aber keine Ziffern) enthalten, komprimieren und wieder dekomprimieren kann. Ihr Filterprogramm muss folgende Aufrufmöglichkeiten von der Kommandozeile aus bieten (die Metasymbole [...] und ...|... stehen für Option bzw. Alternativen):

```
RLE [ -c | -d ] [inFile] [outFile]
```

Bedeutung der Parameter:

-c	die Eingabedatei soll komprimiert werden (Standardannahme),
-d	die Eingabedatei soll dekomprimiert werden,
inFile	Name der Eingabedatei, sonst Standardeingabe (<i>input</i>) und
outFile	Name der Ausgabedatei, sonst Standardausgabe(<i>output</i>).

Beispiel:

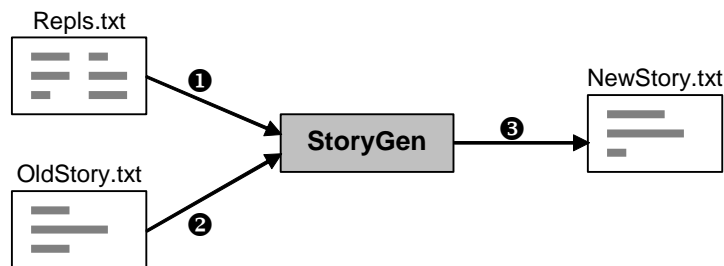


2. Geschichten vom ...

(14 Punkte)

Heutzutage muss ja alles schnell gehen ;-) Wer hat schon noch die Muse, sich der Jahreszeit entsprechende Geschichten für die kleinen und großen Kinder auszudenken. Gesucht ist deshalb ein Pascal-Programm *StoryGen* (als Abkürzung für *story generator*), das es ermöglicht, z. B. aus einer Geschichte vom Osterhasen, die gerade noch aktuell war, schnell eine Geschichte vom Krampus oder vom Christkind zu machen, indem spezielle Wörter ausgetauscht werden (z. B. Osterhase durch Christkind und Ostern durch Weihnachten).

Damit man *StoryGen* flexibel anwenden kann, müssen die durchzuführenden Ersetzungen in einer Textdatei (z. B. *Repls.txt* für *Replacements*) definiert sein und die alte Geschichte in einer zweiten Textdatei (z. B. *OldStory.txt*) stehen. Daraus kann die neue Geschichte in einer dritten Textdatei (z. B. *NewStory.txt*) erzeugt werden, so wie in folgender Abbildung dargestellt:



StoryGen muss zuerst die Ersetzungen einlesen und in einer geeigneten Datenstruktur speichern. Dann muss die alte Geschichtendatei zeilenweise gelesen, alle Ersetzungen in der aktuellen Zeile vorgenommen und die geänderte Zeile in die neue Geschichtendatei geschrieben werden. Die Datei mit den Ersetzungen soll beliebig viele Ersetzungen aufnehmen können, in jeder Zeile aber nur eine (mit der Syntax *oldWord newWord*) enthalten. Eine rudimentäre Ersetzungsdatei für den Übergang von Ostern nach Weihnachten könnte dann folgendermaßen aussehen:

```
Osterhase Christkind
Ostern Weihnachten
...
```



StoryGen muss von der Kommandozeile mit drei Parametern (den Dateinamen der drei beteiligten Textdateien) versorgt werden, so dass es für obiges Beispiel wie folgt aufgerufen werden kann:

```
StoryGen Repls.txt OldStory.txt NewStory.txt
```

Zum Testen finden Sie in *Ostern.txt* eine Geschichte vom Osterhasen. Machen Sie daraus eine möglichst „gute“ Weihnachtsgeschichte.

Übung 3

Aufgabe 1

Lösungsidee

Bei der Dekompression und der Kompression wird Zeichen für Zeichen überprüft ob es eine Nummer oder ein anderer Charakter ist. Falls eine Nummer bei der Dekompression vorkommt, wird das vorherige Zeichen dementsprechend oft eingefügt. Bei der Kompression wird die Anzahl des nacheinander auftretenden Buchstabens gezählt und mit dem Buchstaben in die txt Datei geschrieben. Bei der Kompression und anschließender Dekompression kann eine Zahl bei Sonderkombinationen zu einem falschen Ergebnis führen. Bsp: "aaaa1" würde bei einer Umwandlung "a41" werden. Bei anschließender Dekompression werden jedoch daraus 41 a und nicht wie vorher 4 a Zeichen.

```

1 PROGRAM RLE;
  USES
3   (* sysutils for StrToInt, IntToStr and FileExists *)
  Crt, sysutils;
5
6   (* check if char is a number
7    returns true if number *)
  FUNCTION IsNumber(c: CHAR): BOOLEAN;
9 BEGIN
10  IF ((Ord(c) >= 48) AND (Ord(c) <= 57)) THEN
11    IsNumber := TRUE
12  ELSE
13    IsNumber := FALSE;
14 END;
15
16 (* Get line of TEXT file
17 returns false if EOF, true if not *)
  FUNCTION GetLine(VAR txt: TEXT; VAR curLine: STRING): BOOLEAN;
19 BEGIN
20  IF NOT Eof(txt) THEN BEGIN
21    ReadLn(txt, curLine);
22    GetLine := TRUE;
23  END
24  ELSE
25    GetLine := FALSE; (* EOF of txt *)
26 END;
27
28 (* compresses a string *)
  PROCEDURE CompressString(VAR curLine: STRING);
  VAR
31    s      : STRING;
    curChar : CHAR;
33    count, i, i2 : INTEGER;
  BEGIN
35    curChar := curLine[1];
    curLine := curLine + ' ';
37    s := '';
    count := 0;
39
40    FOR i := 1 TO Length(curLine) DO BEGIN
41

```

```

43  IF curLine[i] = curChar THEN count := count + 1
    ELSE
45      IF count < 3 THEN BEGIN
          FOR i2 := 1 TO count DO s := s + curChar;
          count := 1;
          curChar := curLine[i];
          END
49      ELSE BEGIN
          s := s + curChar + IntToStr(count) ;
          count := 1;
          curChar := curLine[i];
53      END;
    END;

55  IF Length(curLine) > 1 THEN curLine := s;
57  END;

59  (* decompress a string *)
    PROCEDURE DecompressString(VAR curLine: STRING);
61  VAR
        s, temp : STRING;
63      i, i2, count : INTEGER;
        curChar : CHAR;
65  BEGIN
        s := "";
67      temp := "";
        curChar := curLine[1];
69      i := 1;

71      WHILE i <= Length(curLine) DO BEGIN

73          IF NOT IsNumber(curLine[i]) THEN BEGIN
              s := s + curLine[i];
              curChar := curLine[i];
              END
77          ELSE BEGIN
              i2 := i;
79              WHILE IsNumber(curLine[i]) AND (i2 <= Length(curLine)) DO BEGIN
                  temp := temp + curLine[i];
                  i := i + 1;
                  END;
83              i := i - 1;

85              IF temp <> "" THEN count := StrToInt(temp);
                  temp := "";

87              FOR i2 := 1 TO count - 1 DO s := s + curChar;
89              END;
              i := i + 1;
91          END;

93          IF Length(curLine) > 1 THEN curLine := s;
          END;

95  (* compress txt file *)
    PROCEDURE Compress(VAR txt1, txt2 : TEXT);
97  VAR
99      curLine: STRING;

```

```

BEGIN
101  WHILE GetLine(txt1, curLine) DO BEGIN
      CompressString(curline);
103      WriteLn(txt2, curLine);
      END;
105  WriteLn('Compressed');
END;

107
(* decompress txt file *)
109  PROCEDURE Decompress(VAR txt1, txt2 : TEXT);
VAR
111  curLine: STRING;
BEGIN
113  WHILE GetLine(txt1, curLine) DO BEGIN
      DecompressString(curline);
115      WriteLn(txt2, curLine);
      END;
117  WriteLn('Decompressed!');
END;

119
(* check for command line args
121  calls Decompress or Compress *)
PROCEDURE ParamCheck();
VAR
123  option, inFileName, outFileName : STRING;
125  txt1, txt2 : TEXT;  (* text files *)

127  BEGIN
      IF (ParamCount < 3) OR ((ParamStr(1) <> '-c') AND (ParamStr(1) <> '-d')) OR
129      (NOT FileExists(ParamStr(2))) OR (NOT FileExists(ParamStr(3))) THEN
          BEGIN
131              WriteLn('Wrong input, try again: ');
              Write('-c | -d > ');
133              ReadLn(option);

135              Write('inFile > ');
              ReadLn(inFileName);

137              Write('outFile > ');
139              ReadLn(outFileName);
          END
141      ELSE BEGIN
          option := ParamStr(1);
          inFileName := ParamStr(2);
          outFileName := ParamStr(3);
145      END;

147  (*$I-*)

149  (* File initialization *)
      Assign(txt1, inFileName);
151  Reset(txt1);  (* read file *)
      Assign(txt2, outFileName);
153  Rewrite(txt2); (* Rewrite new file or write*)

155  IF IOResult <> 0 THEN
      BEGIN
157          WriteLn('Error opening file!');

```

```

159   Exit;
      END;

161   IF option = '-c' THEN Compress(txt1, txt2) ELSE IF option = '-d' THEN Decompress(txt1, txt2);

163   (* Close Files *)
      Close(txt1);
165   Close(txt2);

167   END;

169 BEGIN

171   ParamCheck();

173 END.

```

rle.pas

```

C:\WINDOWS\system32\cmd.exe
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\2. Semester\AUD\Uebung 3>rle.exe -c d.txt c
.txt
Compressed

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\2. Semester\AUD\Uebung 3>rle.exe -d c.txt d
.txt
Decompressed!

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\2. Semester\AUD\Uebung 3>rle.exe
Wrong input, try again:
-c | -d > -c
inFile > d.txt
outFile > c.txt
Compressed

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\2. Semester\AUD\Uebung 3>

```

Abbildung 1: RLE Optionen

Zu sehen sind die zwei verschiedenen Optionen und die Eingabeaufforderung falls nichts angegeben wurde.

```

1 Hallo dasssssss ist einnnn Tesst.
  AAAAABBBBBbbbbbbssdd!"Ä$%%&/()

```

d.txt

```

2 Hallo das7 ist ein4 Tesst.
  A5B5b6s3dd!"Ä$%%&/()

```

c.txt

In d.txt ist der dekomprimierte Text, in c.txt der komprimierte.

Aufgabe 2

Lösungsidee

Zuerst werden die Wörter die ersetzt werden sollen eingelesen und in einer Hash Tabelle gespeichert. Jedes Zeichen in der Ostern.txt wird eingelesen und vorkommende Wörter werden überprüft. Der berechnete Hashwert eines Wortes wird mit einem Element in der Hash Tabelle an der Position des Hashwertes verglichen. Falls ein Element mit dem Hashcode existiert und genau dasselbe Wort ist (Vergleich, falls ein Wort mit dem selben Hashwert existiert jedoch nicht Zeichen für Zeichen gleich ist), wird das ersetzende Wort in die neue txt Datei eingefügt, andernfalls wird das ursprüngliche Wort eingefügt.

```

PROGRAM storygen;
2  USES
    Crt, sysutils; (* sysutils for filesexits *)
4
CONST
6  EF = CHR(0);      (*END of file character*)
    maxWordLen = 30;  (*max. number of characters per word*)
8  chars = ['a'..'z', 'ä', 'ö', 'ü', 'ß',
    'A'..'Z', 'Ä', 'Ö', 'Ü'];
10 size = 100;

12 TYPE
    Word = STRING[maxWordLen];
14
    NodePtr = ^Node;
16    Node = RECORD
        key: STRING;
18        replacement : STRING;
        next: NodePtr;
20    END; (*Record*)

22    ListPtr = NodePtr;
    HashTable = ARRAY[0..size-1] OF ListPtr;
24
VAR
26    curLine : STRING;      (*current line from file txt*)
    curCh: CHAR;            (*current character*)
28    curLineNr: INTEGER;    (*current line number*)
    curColNr: INTEGER;      (*current column number*)
30    mode : (fillHashTableMode, replaceMode);
    replTXTFile, inTXTFile, outTXTFile : TEXT; (* text files *)
32    newLine : BOOLEAN;

34    (* Sets everything in the ht to NIL *)
PROCEDURE InitHashTable(VAR ht: HashTable);
36 VAR
    h: Integer;
38 BEGIN
    FOR h := 0 TO size - 1 DO BEGIN
40        ht[h] := NIL;
        END;
42    END;

44    (* Create a new Node *)

```

```

FUNCTION NewNode(w, wr : Word; next: NodePtr) : NodePtr;
46 VAR
    n: NodePtr;
48 BEGIN
    New(n);
50 n^.key := w;
    n^.replacement := wr;
52 n^.next := next;

54 NewNode := n;
END;

56 (* compiler hashcode
57 returns hashcode of string *)
58 FUNCTION HashCode(key: String): Integer;
60 BEGIN
    IF Length(key) = 1 THEN
62 HashCode := (Ord(key[1]) * 7 + 1) * 17 MOD size
    ELSE
64 HashCode := (Ord(key[1]) * 7 + Ord(key[2]) + Length(key)) * 17 MOD size
END;

66 (* Lookup combines search and prepend *)
68 FUNCTION Lookup(w, wr : Word; VAR ht: HashTable) : NodePtr;
    VAR
70 i: Integer;
    n: NodePtr;
72 BEGIN
    i := HashCode(w);
74 n := ht[i];

76 WHILE (n <> NIL) AND (n^.key <> w) DO BEGIN
    n := n^.next;
78 END;

80 IF n = NIL THEN BEGIN
    n := NewNode(w, wr, ht[i]);
82 ht[i] := n;
    END;
84 Lookup := n;
END;

86 (* Searches for a string in the hashtable
87 returns NIL if not found *)
88 FUNCTION Search(key: String; ht: HashTable) : NodePtr;
90 VAR
    i: Integer;
92 n: NodePtr;
94 BEGIN
    i := HashCode(key);
    n := ht[i];
96
98 WHILE (n <> NIL) AND (n^.key <> key) DO BEGIN
    n := n^.next;
    END;
100 Search := n;
END;
102

```



```

104  (* updates curChar *)
104  PROCEDURE GetNextChar(VAR txt: TEXT);
104  BEGIN
106    IF curColNr < Length(curLine) THEN BEGIN
106      curColNr := curColNr + 1;
108      curCh := curLine[curColNr];
108    END
110    ELSE IF (curColNr = 0) AND (curColNr >= Length(curLine)) THEN BEGIN
112      CASE mode OF
112        replaceMode: BEGIN
114          WriteLn(outTXTFile);
114          ReadLn(txt, curLine);
116          curLineNr:= curLineNr + 1;
116          curColNr := 0;
118          curCh := ' '; (* separate lines by ' ' *)
118        END;
120      END;
120    END
122    ELSE BEGIN (* curColNr >= Length(curLine) *)
122      IF NOT Eof(txt) THEN BEGIN
124        CASE mode OF
124          replaceMode: BEGIN
126            newLine := TRUE;
126          END;
128          END;
128        END;
130        ReadLn(txt, curLine);
132        curLineNr:= curLineNr + 1;
132        curColNr := 0;
134        curCh := ' '; (* separate lines by ' ' *)
134      END
136      ELSE
136        curCh := EF;
138      END;
138    END; (* GetNextChar *)
140
140  (* Creates word from char
142    – mode decides if between reading or writing
142    returns the next word in a string *)
144  PROCEDURE GetNextWord(VAR w: Word; VAR lnr: INTEGER; VAR txt: TEXT);
144  BEGIN
146    WHILE (curCh <> EF) AND NOT (curCh IN chars) DO BEGIN
146      CASE mode OF
146        replaceMode: BEGIN
148          IF NOT (curCh IN chars) THEN Write(outTXTFile, curCh);
148        END;
150        END;
150      END;
152      GetNextChar(txt);
152    END;
154
154    lnr := curLineNr;
156    IF curCh <> EF THEN BEGIN
156      w := curCh;
158      GetNextChar(txt);
158
160      WHILE (curCh <> EF) AND (curCh IN chars) DO BEGIN

```

```

162     w := Concat(w, curCh);
        GetNextChar(txt);
164     END;
166     ELSE
        w := '';
168     END; (* GetNextWord *)

(* Fills the HashTable with the replacements *)
170 PROCEDURE FillHashTable(VAR ht: HashTable; VAR txt: TEXT);
VAR
172     w, wr: Word;    (*current word*)
    lnr: INTEGER;    (*line number of current word*)
174     n: LONGINT;    (*number of words*)

176 BEGIN
    curLine := '';
178     curLineNr := 0;
    curColNr := 1;    (*curColNr > Length(curLine) FORces reading of first line*)
180     GetNextChar(txt); (*curCh now holds first character*)
    n := 0;
182     mode := fillHashTableMode;
    w := '';

184
    GetNextWord(w, lnr, txt);
186     GetNextWord(wr, lnr, txt);
    WHILE Length(w) > 0 DO BEGIN
188         IF (w <> '') AND (wr <> '') THEN Lookup(w, wr, ht);
            n := n + 1;
190         GetNextWord(w, lnr, txt);
            GetNextWord(wr, lnr, txt);
192     END;
    WriteLn('Found ', n, ' replacements');
194 END;

(* Replaces a word if there is a replacement *)
196 PROCEDURE Replace(ht: HashTable);
VAR
198     w : Word;    (*current word*)
    lnr: INTEGER;    (*line number of current word*)
200     n: LONGINT;    (*number of words*)
    temp : NodePtr;

202
204 BEGIN
    curLine := '';
206     curLineNr := 0;
    curColNr := 1;    (*curColNr > Length(curLine) forces reading of first line*)
208     GetNextChar(inTXTFile); (*curCh now holds first character*)
    n := 0;
210     mode := replaceMode;
    w := ' ';

212
    WriteLn('Replacing... ');

214
    GetNextWord(w, lnr, inTXTFile);
216     WHILE Length(w) > 0 DO BEGIN
        (* Check if word is the word from the ht, and not just a word
218         with the same hashcode *)

```

```

temp := Search(w, ht);
220
IF newLine THEN BEGIN
222   WriteLn(outTXTFile);
   newLine := FALSE;
224 END;

IF temp <> NIL THEN BEGIN
226   //WriteLn('w: ', w, ' temp: ', temp^.key, ' temp repl.: ', temp^.replacement, ' ');
228   IF (temp^.key = w) THEN Write(outTXTFile, temp^.replacement)
   ELSE Write(outTXTFile, w);
230 END
ELSE
232   Write(outTXTFile, w);

n := n + 1;
  GetNextWord(w, lnr, inTXTFile);
236 END;
  WriteLn('Finished!');
238 END;

(* check for command line args
   calls Decompress or Compress *)
242 PROCEDURE ParamCheck();
VAR
244   replaceFileName, inFileName, outFileName : STRING;
   ht : HashTable;
246
BEGIN
248   IF (ParamCount < 3) OR (NOT FileExists(ParamStr(2))) OR
   (NOT FileExists(ParamStr(3))) THEN
250     BEGIN
       WriteLn('Wrong input, try again: ');
252       Write('replaceFile > ');
       ReadLn(replaceFileName);

254
       Write('inFile > ');
256       ReadLn(inFileName);

258       Write('outFile > ');
       ReadLn(outFileName);
260     END
   ELSE BEGIN
262     replaceFileName := ParamStr(1);
     inFileName := ParamStr(2);
264     outFileName := ParamStr(3);
   END;

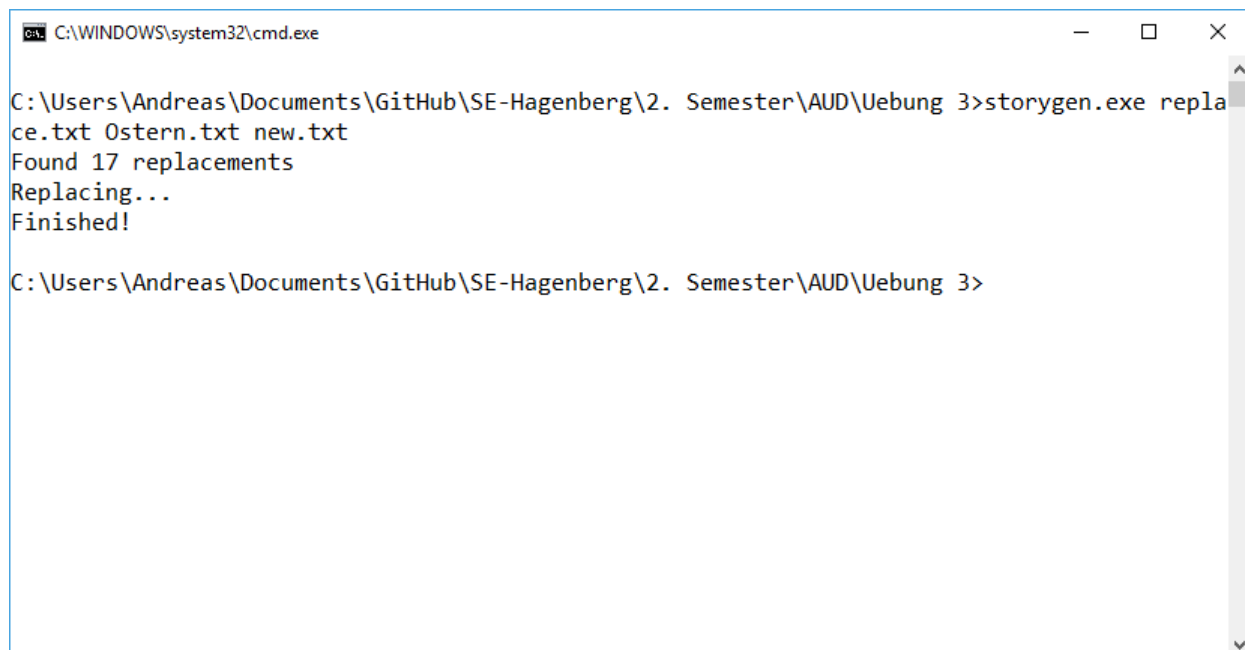
266 InitHashTable(ht);

268
(*$I—*)
270 (* File initialization *)
  Assign(replTXTFile, replaceFileName);
272 Reset(replTXTFile); (* read file *)
  Assign(inTXTFile, inFileName);
274 Reset(inTXTFile); (* read file *)
  Assign(outTXTFile, outFileName);
276 Rewrite(outTXTFile); (* Rewrite new file or write*)

```

```
278 (* Check IOResult for opening errors *)  
    IF IOResult <> 0 THEN  
280 BEGIN  
    WriteLn('Error opening file!');  
282 Exit;  
    END;  
284  
    (* closing repl. cause we dont need it anymore *)  
286 FillHashTable(ht, replTXTFile);  
    Close(replTXTFile);  
288  
    Replace(ht);  
290  
    (* close files *)  
292 Close(inTXTFile);  
    Close(outTXTFile);  
294  
    END;  
296 BEGIN  
298 ParamCheck();  
300 END.
```

storygen.pas



```
cmd. C:\WINDOWS\system32\cmd.exe  
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\2. Semester\AUD\Uebung 3>storygen.exe repla  
ce.txt Ostern.txt new.txt  
Found 17 replacements  
Replacing...  
Finished!  
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\2. Semester\AUD\Uebung 3>
```

Abbildung 2: Storygen

```

1 Osterhasen Weihnachtsmann
  Osterhase Weihnachtsmann
3 Ostereier Geschenke
  Eier Geschenke
5 Ei Geschenk
  Hähnchen Elfen
7 Hühnchen Elfen
  Huhn Elfe
9 Henne Elfe
  Küken Elfen
11 Farbe Geschenkpapier
   Farbtöpfe Geschenkpapierrollen
13 Henne Elfe
   Osterfest Weihnachtsfest
15 Blüten Verpackung
   Blumen Geschenke
17 bemalt verpackt

```

`replace.txt`

In der `replace.txt` werden die Wörter die ersetzt werden sollen und die Wörter durch die sie ersetzt werden sollen gespeichert, getrennt durch ein beliebiges Trennzeichen (außer Buchstaben).

```

1 Die Geschichte vom Osterhasen Hanni!

3 Wißt Ihr liebe Kinder, vor langer, langer Zeit gab es noch keinen Osterhasen.
  Da legten die Hühnchen die Eier und die Hähnchen bemalten sie. In
5 einem Körbchen trugen sie vorsichtig die Eier zu den Kindern und
  stellten es vor die Tür. Die Kinder warteten schon gespannt auf das
7 Osterfest. Und wenn es dann endlich so weit war, sprangen sie schnell
  aus ihren Betten und öffneten neugierig die Tür. Wie freuten sie sich über

```

`Ostern.txt`

```

1 Die Geschichte vom Weihnachtsmann Hanni!

2 Wißt Ihr liebe Kinder, vor langer, langer Zeit gab es noch keinen Weihnachtsmann.
4 Da legten die Elfen die Geschenke und die Elfen bemalten sie.
  In einem Körbchen trugen sie vorsichtig die Geschenke zu den Kindern
6 und stellten es vor die Tür. Die Kinder warteten schon gespannt auf
  das Weihnachtsfest. Und wenn es dann endlich so weit war, sprangen sie
8 schnell aus ihren Betten und öffneten neugierig die Tür. Wie freuten sie sich

```

`new.txt`

Zu sehen sind die ersten 8 Zeilen der `Ostern.txt` und der `new.txt`. Wörter, die in der `replace.txt` vorgegeben sind, wie Osterhasen oder Hühnchen wurden durch die in der `replace.txt` vorhandenen Wörtern ersetzt.