1. Bedienhinweise

1.1 Start von TEXTSTAT

Das Programm kann unter Microsoft DOS über die Kommandozeile oder unter Microsoft Windows in der MS-DOS-Eingabeaufforderung bzw. Eingabeaufforderung[1] mit `textstat datei.*`[2] aufgerufen werden.

1.2 Parameter

Nachfolgend werden die Parameter dargestellt, die beim Programmaufruf angegeben werden können.

```
TEXTSTAT Version 1.0 Build 1019, Copyright (C) 2001 Torsten Weber

TEXTSTAT [[Laufwerk:]Pfad] Datei [D]

[Laufwerk:]Pfad]  - Bezeichnet das Verzeichnis mit Datei

[D]               - MS-DOS Text

                    Ist dieser Parameter nicht angegeben, wird der Windows-
                    Standard-Zeichensatz verwendet. Der Parameter hat Aus-
                    wirkung auf die Berechnung der Anzahl und Häufigkeiten
                    der im Text vorkommenden deutschen Umlaute.
```

Wird keine Datei angegeben, gibt das Programm eine entsprechende Meldung aus.

ABBILDUNG 1: MELDUNG VON TEXTSTAT

1. Unter Microsoft Windows NT / 2000 ist dies sowohl in der Eingabeaufforderung („cmd.exe“) als auch in der MS-DOS-Eingabeaufforderung („command.com“) möglich.
2. Es werden alle Dateiendungen unterstützt, z. B. „text.txt“ oder „text.pl“.

Wird eine große Datei eingelesen, gibt das Programm eine Meldung („Bitte warten...“) aus. Nach dieser kann der Benutzer durch eine entsprechende Menüwahl[1] die Anzahl der Buchstaben, Sätze oder die Häufigkeiten der einzelnen Buchstaben (Buchstabe, Anzahl, prozentuale Verteilung) anzeigen. Das Programm berechnet die Häufigkeiten aufgrund aller im Text vorkommenden Buchstaben, also: Buchstaben + Umlaute. TEXTSTAT kann durch entsprechende Menüwahl jederzeit beendet werden.

ABBILDUNG 2: EINFACHE BEDIENUNG DURCH TASTENANSCHLAG

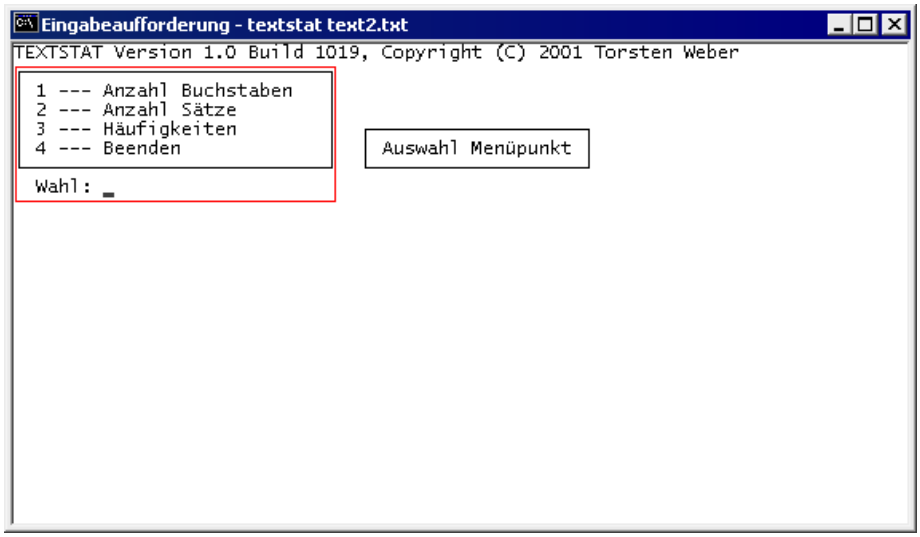
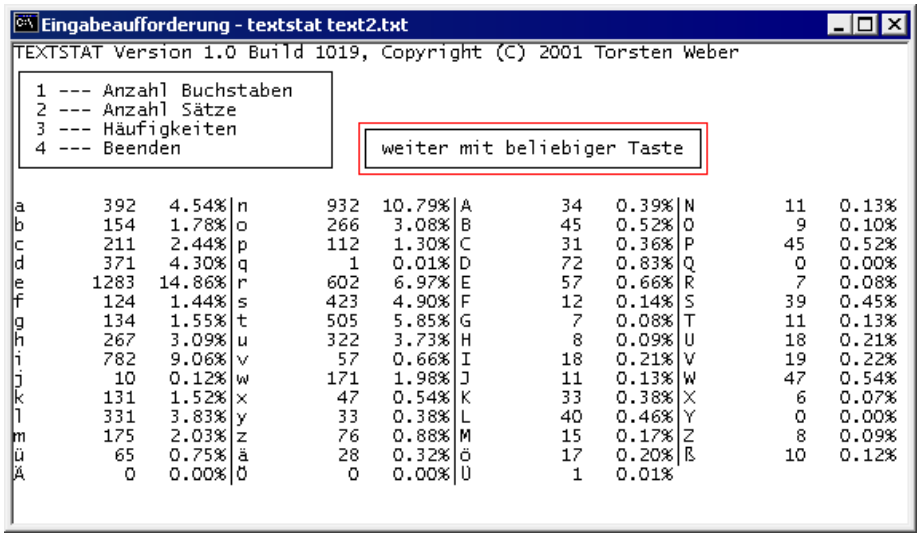


ABBILDUNG 3: EINFACHE BEDIENUNG DURCH TASTENANSCHLAG



1. Tastenanschlag

2. Testläufe

2.1 Beschreibung der Testumgebung

Für die Testläufe wurde, soweit nicht anders angegeben, ein Pentium-II-350 MHz System mit 352 MB RAM, verwendet. Die Testläufe mit den Dateien „text1.txt“, „text2.txt“, „text3.txt“, „text5.txt“, „text6.txt“, „dos.txt“, „windows.txt“ und „abk.dat“ fanden ausschließlich unter dem Betriebssystem Microsoft Windows 2000[1] statt.

Mit der Datei „text4.txt“ fanden Testläufe unter den Betriebssystemen

- Microsoft Windows 95
- Microsoft Windows 98
- Microsoft Windows ME
- Microsoft Windows NT 4.0

statt.

2.2 Beschreibung der Testdaten

Für die Testdaten wurden Textdateien mit einem Texteditor erstellt. Bei einigen Dateien wurde der Inhalt durch mehrmaliges Kopieren von Textpassagen hergestellt. Es wurde, bis auf Test 3 und Test 10, nur auf Buchstaben des ASCII-Zeichensatzes und auf deutsche Umlaute[2] zurückgegriffen.

ABBILDUNG 4: AUSZUG AUS EINER TESTDATEI

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄ
ÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääö
ßÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZü
ääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXY
ZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVW
XYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTU
VWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRS
TUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQ
RSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNO
PQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLM
NOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJK
LMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHI
JKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFG
HIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDE
FGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABC
DEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzA
BCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxy
zABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvw
xyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstu
vwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrs
tuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopq
rstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmno
pqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklm
nopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijk
lmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghi
jklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefg
hijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcde
fghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabc
defghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖa
bcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜ
ÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääß
ÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüä
ääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
üääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWX
YZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUV
WXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüääößÄÜÖabcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRST
```

1. Soweit nicht anders angegeben, wurde nicht die Option „D“ (MS-DOS Text) verwendet.
2. Windows-Standard-Zeichensatz

3. Sicherstellung der Erfüllung der Aufgabenstellung

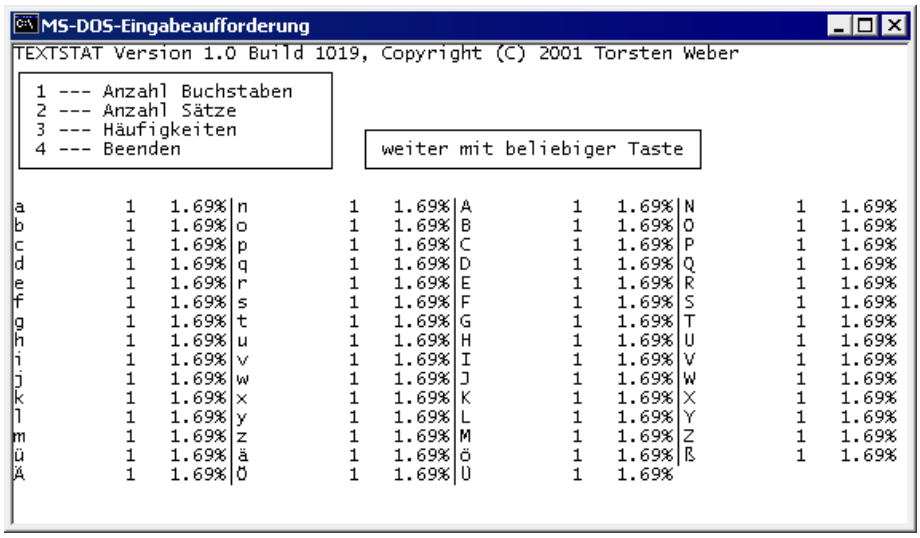
3.1 Test 1 - Textdatei „text1.txt“

Mittels diesem Test sollte geprüft werden, ob das Programm alle Buchstaben und die deutschen Umlaute richtig erkennt und verarbeitet. Entsprechend enthält die Datei „text1.txt“ jeden Buchstaben des deutschen Alphabets bzw. Umlaut nur einmal. Die Gesamtanzahl ergibt sich also aus 26 * 2 = 52 Buchstaben (+ 7 Umlaute).
Die Häufigkeiten ergeben sich aus 1 * 100 / (26 * 2 + (7)) = 1,694 %.

ABBILDUNG 5: INHALT DER DATEI „TEXT1.TXT“

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüäößÄÜÖ
```

ABBILDUNG 6: AUSWERTUNG MIT TEXTSTAT



Wie sich aus Abbildung 6 erkennen läßt, wurden alle Buchstaben oder Umlaute ordnungsgemäß erkannt und die Häufigkeiten der einzelnen Buchstaben bzw. Umlaute korrekt berechnet.

3.2 Test 2 - Korrekte Auswertung der Textdatei „text2.txt“

Für diesen Test wurde eine neue Textdatei erstellt, deren Inhalt aus einem Auszug aus einer Veröffentlichung[1] bestand, die online abgefragt wurde. Um die korrekte Berechnung der einzelnen Buchstaben und deutschen Umlaute zu testen, wurde der gleiche Text in Microsoft Word eingefügt und die einzelnen Buchstaben / Umlaute mittels der Funktion „Suchen und Ersetzen“[2] durch die Zahl 1 ersetzt. Durch Microsoft Word wird die Anzahl der ersetzten Buchstaben / Umlaute angegeben. Diese Anzahl mußte zwingend mit der Anzahl übereinstimmen, die von TEXTSTAT ermittelt wurde. Da nach den „Suchen und Ersetzen“-Vorgängen keine Buchstaben bzw. Umlaute im Text mehr vorhanden sein durften, wurde mit der Option „beliebiger Buchstabe“ im Text nochmals gesucht. Richtigerweise gab Microsoft Word die Meldung aus, daß das gesuchte Element - also ein beliebiger Buchstabe - nicht gefunden werden konnte.

ABBILDUNG 7: DATEI „TEXT2.TXT“

```
Die Softwareschmiede Borland ist seit Jahren bei Windows-Entwicklern der Inbegriff für RAD-Tools. Mit Kylix hat das
Unternehmen nun auch eine Entwicklungsumgebung für Linux veröffentlicht, die es vor allem Delphi-Programmierern
erleichtert, Applikationen ohne Reibungsverluste nun auch unter Linux bereit zu stellen. Was für klassische C++-Ent-
wickler, die auf ihren emacs schwören, wie der Untergang des Abendlandes wirken mag, könnte auf einen Schlag Tausende
von Delphianern für die Plattform Linux begeistern. Wir sprachen mit David I, Vice President Developers Relations,
über den Cross-Plattform-Ansatz von Kylix, der ein Bindeglied zwischen der Windows-Welt und dem freien Betriebssystem
bildet und über die weiteren Ambitionen des Unternehmens in Sachen Linux.


Der Entwickler: David, ich freue mich, Sie hier auf der LinuxWorld mit Ihrem neuen Produkt Kylix zu sehen. Warum haben
Sie es letztendlich "Kylix" genannt und nicht "Delphi für Linux"?

David Intersimone: Nun, es ist im Prinzip ein anderes Produkt mit einer anderen Plattform und einem anderen Komponen-
tentreiber. Kylix generiert native Linuxprogramme. Außerdem bringt es die Cross Platform Component Library CLX mit,
die das neue Bindeglied zwischen Linux und Windows darstellt im Hinblick auf diesen Cross-Plattform-Ansatz. Eben weil
es sich um eine andere Plattform und einen anderen Komponententreiber handelt, haben wir uns auch für einen neuen
Namen entschieden. Natürlich machen wir deutlich, dass Kylix zu Delphi kompatibel ist.


Lassen Sie uns über die Component Library CLX sprechen. CLX wird sowohl unter Delphi als auch unter Kylix genutzt wer-
den können?

Delphi 6 wird über VCL, die Visual Component Library für Windows, und über die Cross Platform Library CLX verfügen,
sodass damit sowohl neue VCL-Applikationen als auch CLX-Applikationen entwickelt werden können. Eine andere Möglich-
keit ist es, Linux-Applikationen auf die Windows-Plattform und Delphi zu bringen. Oder umgekehrt können Delphi-Appli-
kationen dann auch auf die Linux-Plattform portiert werden.


Wo liegen die Vorteile für Windows-Entwickler? Warum sollten sie Kylix benutzen?

Nun, aus der Sicht von Windows-Entwicklern bringt Kylix eine Menge Vorteile mit sich. Zunächst einmal können sie Kylix
aus dem Stand heraus benutzen und dabei alle ihre bisherigen Fähigkeiten voll einsetzen. Sämtlicher Code, der bereits
geschrieben wurde, alle Applikationen, die man bereits entwickelt hat und alle Komponenten können unter Kylix weiter
verwendet werden. Das heißt, alle bisherigen Möglichkeiten können ohne Reibungsverluste nun auf einer weiteren Platt-
form ausgespielt werden. Damit eröffnet sich eine wirkliche Freiheit der Wahl. Man muss sich nicht auf eine Plattform
festlegen und sagen, ich will nur für Windows oder nur für Linux entwickeln. Eher ist es so, dass der Entwicklungspro-
zess mit Kylix zwischen beiden Plattformen gewissermaßen hin und her fließen kann.


Sie sind bei Borland als Vice President Developers Relations für die Entwicklerinteressen zuständig. Wie wollen Sie
die Entwickler dabei unterstützen, Kylix einzusetzen? Wie wollen Sie einen Prozess fördern, der die Ressourcen der
Entwickler, die bisher nur unter Delphi zur Verfügung standen, auf Kylix überträgt und damit wohl auch mehr Anwender
für Linux als Plattform interessiert?

Sie sprechen hier verschiedene Aspekte an. Zunächst einmal haben wir schon verschiedene Artikel über Kylix bereit
gestellt und White Papers veröffentlicht. Darin wird erklärt, wie man am besten Delphi-Applikationen mit Kylix auf
Linux portiert. Es gibt bestimmte Windows-spezifische Aspekte, die dabei beachtet werden müssen. Wir erklären den Ent-
wicklern, wie sie bestimmte Features, die wir in Kylix integriert haben, am besten nutzen. Wir erläutern zum Beispiel,
wie spezielle Windows-API- Funktionsaufrufe in Linux-Kernel-Funktionsaufrufe transformiert werden. Bei der Portierung
```

1. www.derentwickler.de, Ausgabe 3.2001 (Mai/Juni), „Ich habe immer meine Borland-T-Shirts getragen“, Bericht von Masoud Kamali
2. Die Option Groß-/Kleinschreibung war aktiviert.

müssen also schon einige systemspezifische Anpassungen vorgenommen werden. Komponenten können aber mit minimalen Veränderungen genutzt werden, das gilt auch für Datenbankzugriffe. Einige Projekte wurden bereits erfolgreich portiert. Ein gutes Beispiel ist ein Chat Server, der über http läuft und in Delphi geschrieben war. Der wurde, obwohl das im Hinblick auf die Sockets einiges an Arbeit bedeutet hat, in 16 Stunden mit Kylix portiert. Über solche Prozesse schreiben wir Artikel, die den Cross-Plattform-Ansatz näher erläutern und verdeutlichen. Auf unserer Community Site tut sich da schon einiges, und natürlich in Magazinen und den ersten Buchpublikationen. Für Leute, die Produkte und Komponenten entwickeln, werden wir natürlich auch Informationen darüber bereit stellen, wer Kylix bereits wie unterstützt.

Welche Editionen wird es von Kylix geben?

Wir bringen drei verschiedene Editionen auf den Markt. Zum einen gibt es eine Desktop Developer Edition, dann eine Server Developer Edition und zudem noch ein Open Source Toolkit. Die Developer Edition beinhaltet alle GUI-Tools, die Entwicklungsumgebung und die Komponenten für die GUI-Entwicklung, die Datenbankkomponenten, zum Beispiel die Treiber für InterBase oder MyBase. Die Server Edition beinhaltet außerdem die Unterstützung von DB2 und Oracle. Das Open Source Toolkit beinhaltet die Entwicklungsumgebung, die Visual Components und die Datenbankkomponenten. Die Lizenz ist eine GPL-Lizenz. Dieses Toolkit soll etwa zur Jahresmitte erscheinen und kann dann im Internet frei heruntergeladen werden, so wie unsere JBuilder Foundation. Für 99 Dollar können die Entwickler das Ganze auf CD bekommen mit entsprechendem Manual. Die Desktop Developer Edition wird für 999 Dollar ausgeliefert. Für Delphi-5-Anwender wird es außerdem Sonderkonditionen geben. Die Server Edition vertreiben wir für 1.999 Dollar. Außerdem soll es noch eine Enterprise Edition geben. Diese Versionen haben bisher noch keine CORBA-, SOAP- und XML-Unterstützung. An diesen Entwicklungen arbeiten wir jedoch schon, denn sie werden Bestandteil von Delphi 6 sein und danach auch in die nächste Kylix-Enterprise-Version einfließen.

Wann können denn die C++-Entwickler mit einem C++Builder für Linux rechnen?

Betrachten Sie einmal unsere Entwicklungs- und Produktpolitik. Wir haben unsere Delphi- und C++-Produkte, die beide dieselbe Umgebung und dieselbe Komponentenbibliothek nutzen. Sie haben verschiedene Compiler und verschiedene Sprachen als Interface zu den Komponenten. Die Entwicklungsprozesse bei Borland finden in Sequenzen statt, denn an den Produkten arbeiten dieselben Entwicklungsteams. Wir liefern eine Version von Delphi für Windows aus und einen C++Builder für Windows. Jetzt haben wir dazwischen Kylix entwickelt. Es gab seit längerer Zeit keine neue Delphi Version, also arbeiten wir nun zuerst einmal an Delphi 6, um es fertig zu stellen. Und danach wird dann sicher auch ein C++Builder für Linux auf den Markt kommen. Damit ist dann etwa in der zweiten Jahreshälfte 2001 zu rechnen.

Welche längerfristigen Perspektiven im Hinblick auf die Windows Unterstützung bieten Sie Delphi-Entwicklern?

Wir unterstützen die Windows-Plattform schon Jahre lang, im Prinzip, seit es sie gibt. Auch im Hinblick auf die .NET-Strategie von Microsoft bleiben wir am Ball. Das ist ein wirklich längerfristiges Projekt, und wir werden auch diese Plattform unterstützen. Sie unterscheidet sich von der bisherigen Windows-Plattform, denn sie hat ein anderes Framework. Das ist für uns keine Entweder-Oder-Entscheidung. Wir sind jetzt einfach auch im Bereich Linux aktiv. Und wir hoffen, auch noch für andere aufstrebende Plattformen, beispielsweise im Mobile Computing oder bei Handhelds, Entwicklungsarbeit zu leisten. Das Windows Business ist nach wie vor ein sehr gutes Geschäftsfeld für uns. Entwickler müssen sich zum Beispiel keine Gedanken darüber machen, was aus der Visual Component Library wird. Wir werden auf jeden Fall in den nächsten Jahren der VCL weitere Komponenten hinzufügen. Der einzige Bereich, in dem wir keine Neuentwicklungen mehr vorantreiben, sondern nur noch Support bieten, ist die Borland Database Engine, die nur unter Windows und nicht unter Linux zur Verfügung steht. Hier haben wir eine neue Architektur, dbExpress. Die Architektur von dbExpress ist wesentlich eleganter und vereinfacht die Integration anderer Datenbanken. Die Borland Database Engine wird natürlich auch in Delphi 6, 7 und 8 zur Verfügung stehen, aber wir entwickeln sie eben nicht mehr weiter. dbExpress dagegen ist eher mit ADO vergleichbar. Und dann haben wir noch InterBase Express. Das werden wir auch alles unter Linux verfügbar machen.

Die letzte Zeit bei Borland war ja sehr turbulent. Welche Bedeutung hatten all die Veränderungen? Nach dem Inprise Intermezzo ist Borland nun wieder Borland, und die neue Parole heißt "Back to the roots". Und Sie selbst sind ja anscheinend auch in bester Stimmung. Wie sehen Sie die Zukunft von Borland?

Für viele Entwickler war Borland nie wirklich von der Bildfläche verschwunden. Und unsere Markenpolitik hat sich ja auch immer auf Borland bezogen. Selbst in den Zeiten, als Borland Inprise hieß, habe ich weiter meine Borland T-Shirts getragen, denn für mich war Borland einfach das, was die Entwickler kannten und wollten. Nun haben wir unseren Namen wieder, und ich denke mal, dass das im Prinzip auch unsere Jahre lange Erfahrung und all die Innovationen, die wir entwickelt haben, sehr deutlich zum Ausdruck bringt. Man sollte aber auch nicht sagen, das alte Unternehmen Borland ist nun wieder da, denn wir haben uns ja einige Zeit nicht unbedingt auf die richtigen Dinge konzentriert. Was mich so glücklich macht, ist die Tatsache, dass wir nun unseren Namen wiederhaben, der ja auch ein Stück Identität repräsentiert. Ein anderer Punkt, der mich sehr zufrieden macht, ist die Unterstützung der Entwickler und Kunden, diese Loyalität zu unseren Produkten. Das Jahr 2000 war für uns ein tolles Jahr, denn wir haben unseren Gewinn gesteigert und sind gewachsen. Wir haben unsere Organisation verbessert. Und wir sind immer noch dabei, das Unternehmen zu verändern. Wir expandieren in manchen Bereichen, beispielsweise mit unserem Developer Services Program. Das ist für uns ein neues Business, das wir in den nächsten Jahren weiter ausbauen werden. Wir wollen noch mehr im Hinblick auf Linux machen. Wir beschäftigen uns künftig mit .NET. Wir werden weiter schauen, was die Entwickler wollen und brauchen, zum Beispiel im Hinblick auf Palm Devices. Also, ich bin wirklich happy!

ABBILDUNG 8: ERSETZEN DER EINZELNEN BUCHSTABEN DURCH MICROSOFT WORD

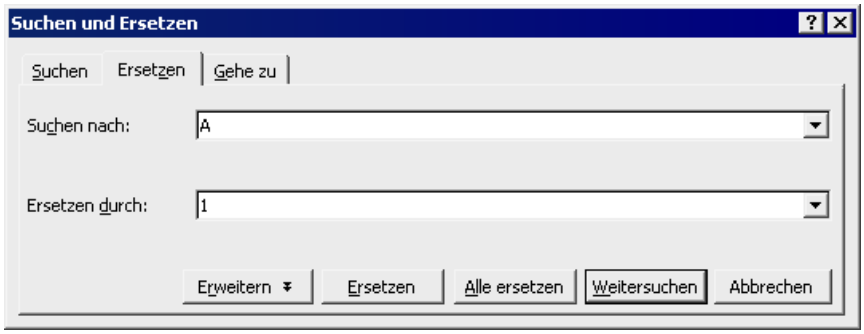


ABBILDUNG 9: ANGABE DER ERSETZTEN BUCHSTABEN / UMLAUTE

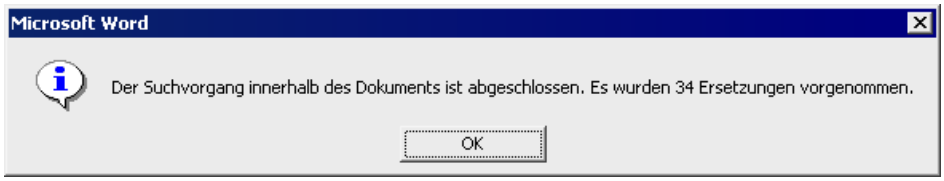


ABBILDUNG 10: AUSWERTUNG MIT TEXTSTAT

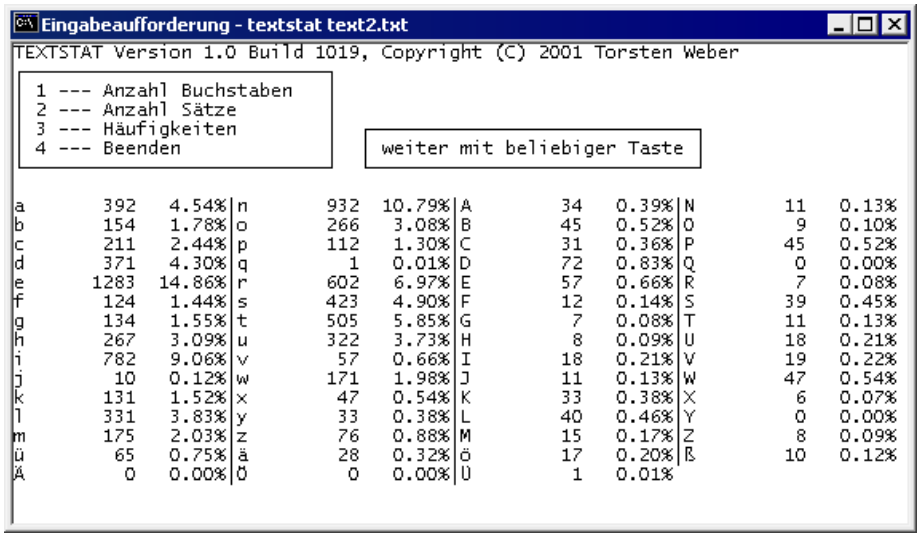
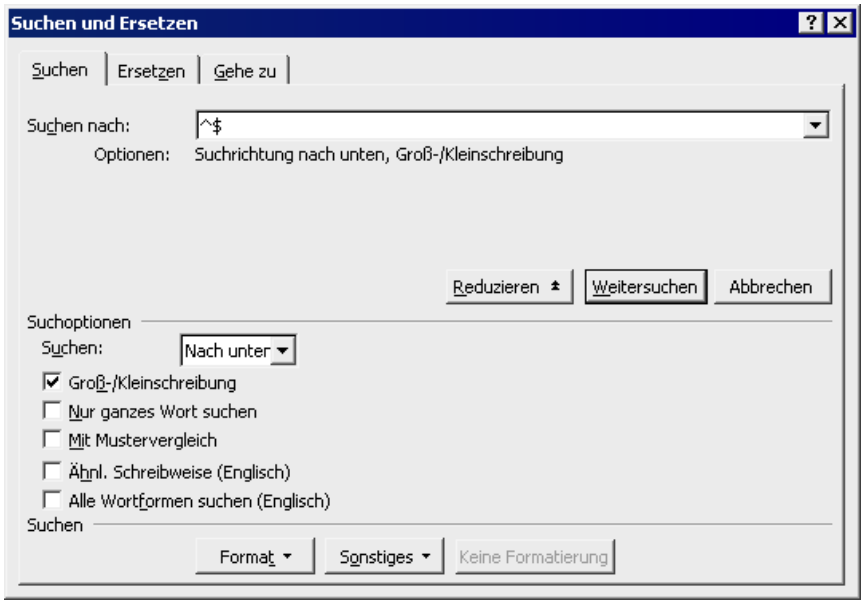


TABELLE 1: AUSWERTUNG MIT MICROSOFT WORD

BUCHSTABE	ANZAHL	BUCHSTABE	ANZAHL
a	392	A	34
b	154	B	45
c	211	C	31
d	371	D	72
e	1.283	E	57
f	124	F	12
g	134	G	7
h	267	H	8
i	782	I	18
j	10	J	11
k	131	K	33
l	331	L	40
m	175	M	15
n	932	N	11
o	266	O	9
p	112	P	45
q	1	Q	0
r	602	R	7
s	423	S	39
t	505	T	11
u	322	U	18
v	57	V	19
w	171	W	47
x	47	X	6
y	33	Y	0
z	76	Z	8
ANZAHL			**8515**
ü	65	Ü	1
ä	28	Ä	0
ö	17	Ö	0
ß	10		
ANZAHL			**121**
GESAMT			**8636**

ABBILDUNG 11: SUCHE NACH BELIEBIGEN BUCHSTABEN



Wie sich aus Abbildung 10 und Tabelle 1 erkennen läßt, wurden alle Buchstaben oder Umlaute ordnungsgemäß erkannt und die Häufigkeiten der einzelnen Buchstaben bzw. Umlaute korrekt berechnet.

3.3 Test 3 - Auswertung einer Binärdatei „text3.txt“

Da TEXTSTAT beim Einlesen der Dateien nicht zwischen Textdateien und Binärdateien unterscheidet und so alle Zeichen außer den Buchstaben des (7-Bit) ASCII-Zeichensatzes bzw. deutschen Umlauten der internationalen IBM-Codetabelle nicht ausgewertet, sollte mit diesem Test nochmals sichergestellt werden, daß das Programm auch beim versehentlichen[1] Einlesen einer Binärdatei nicht in einen undefinierten Zustand gerät.

Dazu wurde die Datei „explorer.exe“ aus dem Verzeichnis „\windows“ des Betriebssystems Microsoft Windows 98 in „text3.txt“ umbenannt und mit TEXTSTAT eingelesen. Der gleiche Test wurde mit einer ca. 113 MB[2] großen Emaildatenbank des Programmes Microsoft Outlook wiederholt.

ABBILDUNG 12: UMBENENNEN DER DATEI „EXPLORER.EXE“

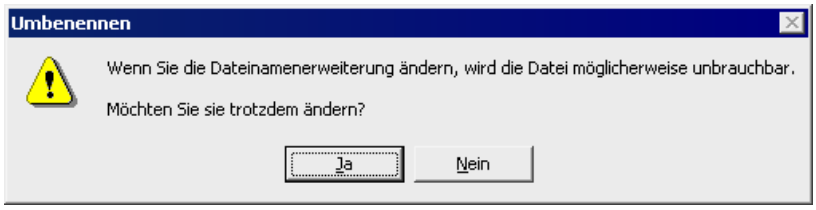
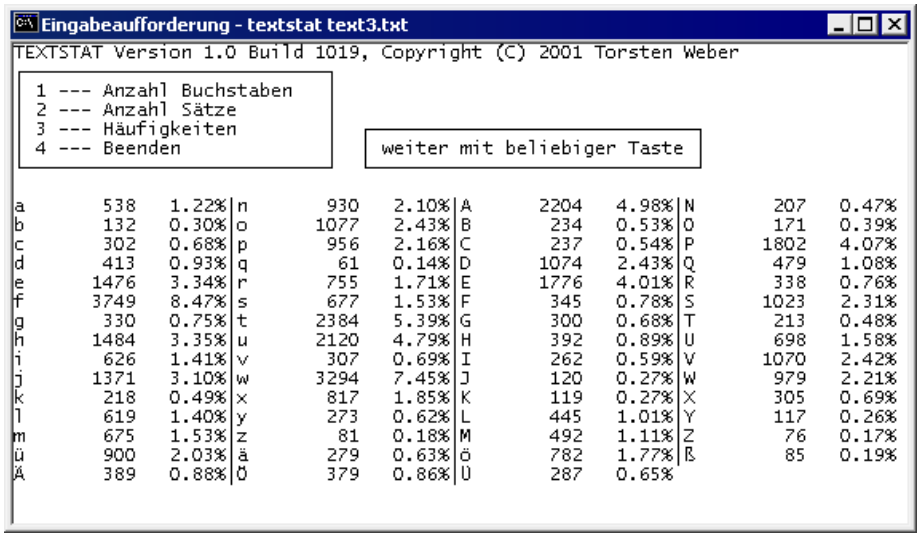


ABBILDUNG 13: AUSWERTUNG MIT TEXTSTAT („EXPLORER.EXE“)



TEXTSTAT hat die Datei „explorer.exe“ als auch die 113 MB große Emaildatenbank ordnungsgemäß eingelesen und keinen undefinierten Zustand eingenommen.

1. TEXTSTAT ist ja gerade zur Analyse von Textdateien wie z. B. „*.txt“, „*.pl“ oder „*.js“ vorgesehen .
2. Die genaue Größe betrug 115.212.288 Bytes.

3.4 Test 4 - Auswertung großer Textmengen in den Textdateien ,,text4.txt“ und ,,text5.txt“

Mittels diesem Test sollte festgestellt werden, wie sich das Programm bei großen Textmengen verhält.
Um einen ungefähren Vergleichswert zu erhalten, wurde mit Microsoft Word 2000 und Adobe FrameMaker 5.5 der gleiche Test durchgeführt. Andere Textverarbeitungsprogramme waren nicht oder nur begrenzt nutzbar, da Absatzgrößen, die größer als 65.536 Zeichen sind, von den meisten Programmen, z. B. Lotus Word Pro oder Corel Draw, nicht unterstützt werden, der Text aber nicht in Absätze unterteilt werden sollte.

Um eine hinreichend große Textmenge zu erhalten, bei der die Häufigkeiten der Buchstaben bzw. Umlaute gleich ist, wurde der unter Abbildung 14 ersichtliche Text n-mal in die Datei „text4.txt“ kopiert, bis die Datei eine Größe von ca. 8,5 MB[1] erreicht hatte. In Adobe FrameMaker und Microsoft Word wurde dieser Text mittels der Zwischenablage einkopiert, die Zeit für einen Seitenumbruch durch die Programme abgewartet und die Statistikanalyse aufgerufen. Die erforderliche Zeit für die Statistikanalyse wurde gemessen.

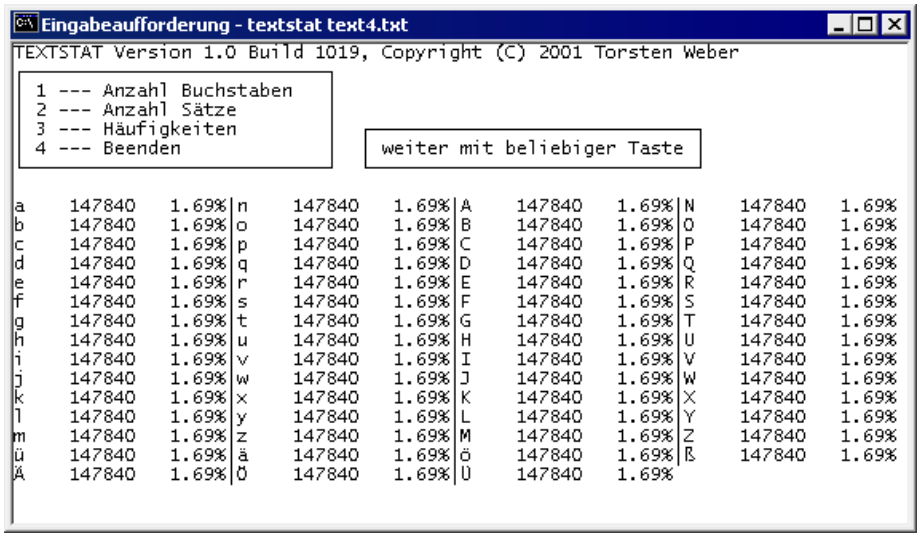
ABBILDUNG 14: N-MAL KOPIERTE TEXTFOLGE DER DATEI ,,TEXT4.TXT“

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZüäößÄÜÖ
```

TABELLE 2: VERGLEICHSWERTE DER GETESTETEN PROGRAMME

PROGRAMMNAME	DAUER BERECHNUNG IN SEKUNDEN	ERRECHNETE BUCHSTABENANZAHL
Adobe Framemaker 5.5	435	8.722.560
Microsoft Word 2000	16	8.722.560
TEXTSTAT 1.0	22	8.722.560

ABBILDUNG 15: AUSWERTUNG MIT TEXTSTAT

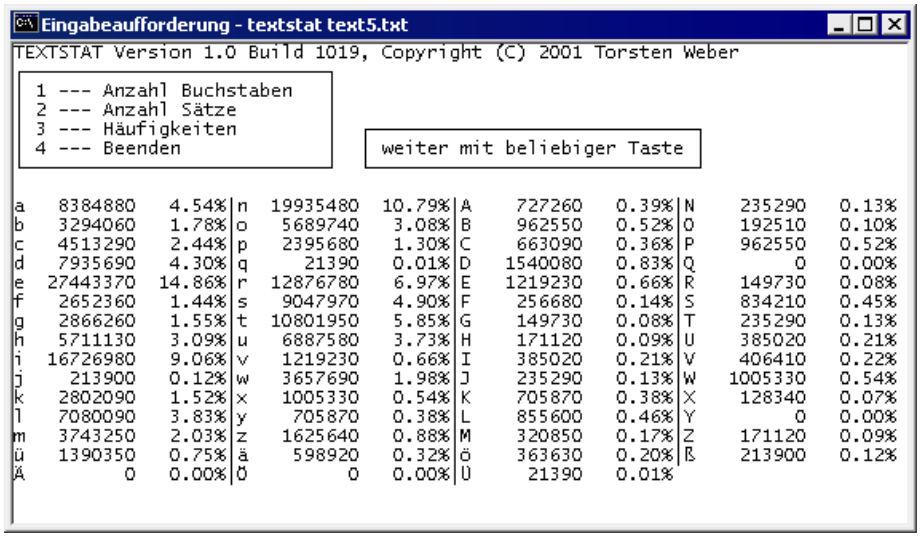


1. Die genaue Größe betrug 8.722.564 Bytes.

Auch wenn ein direkter Vergleich der Meßwerte hier sicherlich nicht sehr sinnvoll ist, wurde aber dennoch gezeigt, daß sich das Programm auch für große Textmengen nicht durch lange Wartezeiten auszeichnet. Alle Programme errechneten die gleiche Buchstabenanzahl.

Um das Verhalten auf extrem große Textmengen zu testen, wurde der Inhalt aus der Datei „text2.txt“ n-mal in die Datei „text5.txt“ kopiert, bis diese eine Größe von ca. 218 MB[1] erreicht hatte. Hier dauerte die Auswertung durch TEXTSTAT ca. 13 Minuten[2]. Da der Inhalt der Datei „text2.text“ n-mal in die Datei „text5.txt“ kopiert wurde, mußte die prozentuale Verteilung - bis auf geringe Abweichungen - gleich sein.

ABBILDUNG 16: AUSWERTUNG MIT TEXTSTAT („TEXT5.TXT“)



Wie aus der Abbildung 10 und Abbildung 16 ersichtlich ist, ist die prozentuale Verteilung der Buchstaben gleichgeblieben. Die Datei wurde also korrekt eingelesen und verarbeitet[3].

1. Die genaue Größe betrug 224.081.801 Bytes.
2. Genau 13 Minuten und 16 Sekunden.
3. Wie leicht bei einem Vergleich der beiden Abbildungen errechenbar ist, wurde der Text aus der Datei „text2.txt“ genau 21.390 Mal in die Datei „text5.txt“ einkopiert.

4. Abhängigkeit vom Entwicklungsrechner / Systemanforderungen

4.1 Test 5 - Auswertung Textdatei „text2.txt“ auf unterschiedlichen Betriebssystemen

Für diesen Test wurde die Datei „text2.txt“ benutzt, bei der ja schon nachgewiesen wurde, daß das Programm die einzelnen Buchstaben / Umlaute korrekt verarbeitet hat. Die Ergebnisse im Test 2 waren folglicherweise auch unter den anderen Betriebssystemen zu erreichen.

- Microsoft Windows 95 bestanden
- Microsoft Windows 98 bestanden
- Microsoft Windows ME bestanden
- Microsoft Windows NT 4.0 bestanden
- Microsoft Windows 2000 bestanden (siehe Test 2)

Es hat sich gezeigt, daß TEXTSTAT unter den oben aufgeführten Betriebssystemen die Ergebnisse richtig berechnet.

4.2 Test 6 - Auswertung Textdatei „text4.txt“

Um zu zeigen, daß TEXTSTAT nur eine geringe Abhängigkeit vom Entwicklungsrechner besitzt, wurde die Auswertung der Datei „text4.txt“ auf mehreren älteren, aber auch neueren Computermodellen nochmals durchgeführt.

TABELLE 3: AUSWERTUNGSZEIT VON TEXTSTAT FÜR DIE DATEI „TEXT4.TXT“

PROZESSOR	RAM IN MB	DAUER IN SEKUNDEN	BETRIEBSSYSTEM
486 SX 33 MHz	4	41	MS Windows 95
Pentium II 350 MHz	352	22	MS Windows 2000
Penitum II 350 MHz	352	26	MS Windows ME
Pentium III 500 MHz	128	20	MS Windows 95
AMD 1000 MHz	256	11	MS Windows 98

Wie aus Tabelle 3 ersichtlich ist, zeigt TEXTSTAT eine relativ geringe Abhängigkeit vom Entwicklungsrechner und ist auch auf älteren Systemen lauffähig.

5. Test der Programmlogik

5.1 Test 7 - Das Erkennen von Abkürzungen mittels der Datei ,,abk.dat“

Die Satzanzahl wird durch TEXTSTAT durch das bloße Zählen der Interpunktionszeichen „?“, „!“ und „.“ ermittelt. Da die Anzahl der Sätze nur dann korrekt berechnet werden kann, wenn aus Sätzen wie

„Das Feuer bzw. Wasser ist lebenswichtig.“

Abkürzungen herausgerechnet werden, sind in der mitgelieferten Datei „abk.dat“ die wichtigsten deutschen Abkürzungen definiert. Da die deutsche Grammatik Situationen liefert, bei denen auch dieser Ansatz fehlläuft, kann dies nur durch ein umfangreiches Wörterbuch bzw. einer Grammtikprüfung verhindert werden, was aber nicht Bestandteil der Aufgabe[1] war.

Deshalb sollte mit diesem Test nur festgestellt werden, daß die definierten Abkürzungen als solche erkannt werden und das Programm die Satzanzahl - unter Berücksichtigung obigen Sachverhalts - richtig berechnet.

5.2 Test 8 - Erkennen von Abkürzungen in der Textdatei ,,abk.dat“

Bei diesem Test wurde die Datei „abk.dat“ eingelesen. Da der Inhalt der Datei ja nur aus Abkürzungen, nämlich den vordefinierten, besteht, durfte als Satzanzahl nur 0 ermittelt werden.

ABBILDUNG 17: AUSWERTUNG MIT TEXTSTAT (,,ABK.DAT“)

Wie aus der Abbildung 17 erkennbar ist, hat das Programm alle Abkürzungen erkannt und die Satzanzahl (0) korrekt berechnet.

1. Aufgabenstellung siehe Anlage 2.

5.3 Test 9 - Erkennen von Abkürzungen in der Textdatei„text2.txt“

Bei diesem Test wurde die Datei „text2.txt“, die schon einmal benutzt wurde, herangezogen. Um einen objektiven Vergleich zu erreichen, ohne die einzelnen Sätze manuell zählen zu müssen, wurde der gleiche Test mit den Programmen Microsoft Word und Corel Draw ebenfalls durchgeführt.

ABBILDUNG 18: AUSWERTUNG MIT TEXTSTAT („TEXT2.TXT“)

ABBILDUNG 19: AUSWERTUNG MIT MICROSOFT WORD („TEXT2.TXT“)

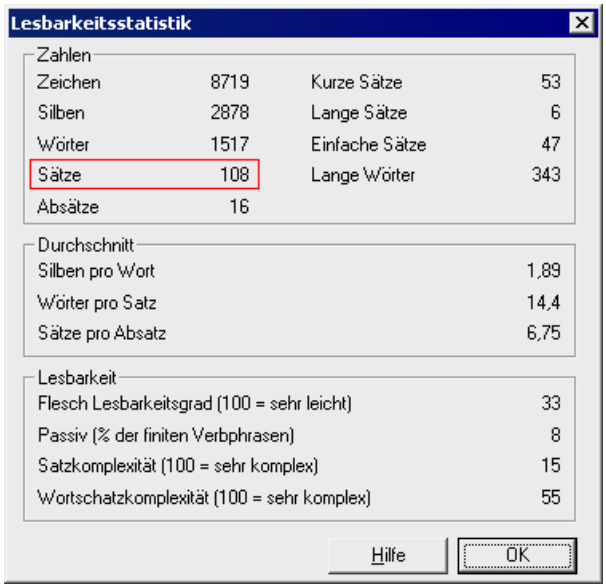
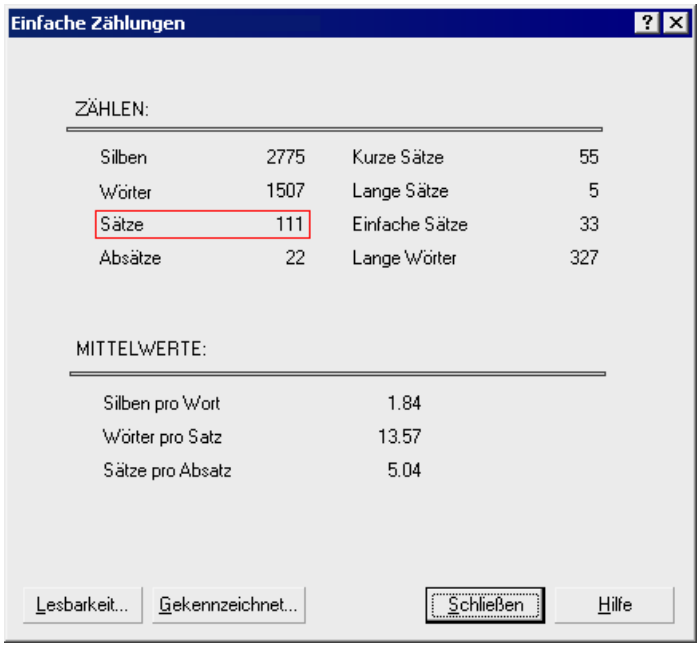


ABBILDUNG 20: AUSWERTUNG MIT COREL DRAW („TEXT2.TXT“)



Wie aus der Abbildung 18, Abbildung 19 und Abbildung 20 ersichtlich ist, weichen die Ergebnisse um einen bzw. zwei Sätze ab. Aus schon erwähnten Gründen ist diese Abweichung vertretbar, da selbst kommerzielle Programme meist recht unterschiedliche Ergebnisse ermitteln.

5.4 Test 10 - Verwendung unterschiedlicher Zeichensätze in ,,dos.txt“ und ,,windows.txt“

Da die Zeichensätze, die in den Textdateien verwendet werden, unterschiedlich sein können, kann vom Benutzer angegeben werden, nach welchem Zeichensatz (MS-DOS Text oder Windows-Standard-Zeichensatz) ausgewertet werden soll.

Gibt der Benutzer keine Option „D“ beim Programmaufruf an, wird nach Windows-Standard-Zeichensatz ausgewertet.

Bei der Auswertung muß TEXTSTAT diese Zeichensätze auseinanderhalten können und dem Benutzer die, entsprechend seiner Wahl, richtigen Ergebnisse anzeigen.

Dazu wurden die Dateien „dos.txt“ und „windows.txt“ angelegt. Ihr Inhalt war identisch und bestand aus den Zeichen:

1 Mal ü
2 Mal ä
3 Mal ö
4 Mal ß
5 Mal Ä
6 Mal Ö
7 Mal Ü.

Die Datei „dos.txt“ wurde mit dem MS-DOS Editor, die Datei „windows.txt“ mit Microsoft Notepad erstellt.

ABBILDUNG 21: ERSTELLUNG DER DATEI ,,NOTEPAD.TXT“ MIT DEM MS-DOS EDITOR

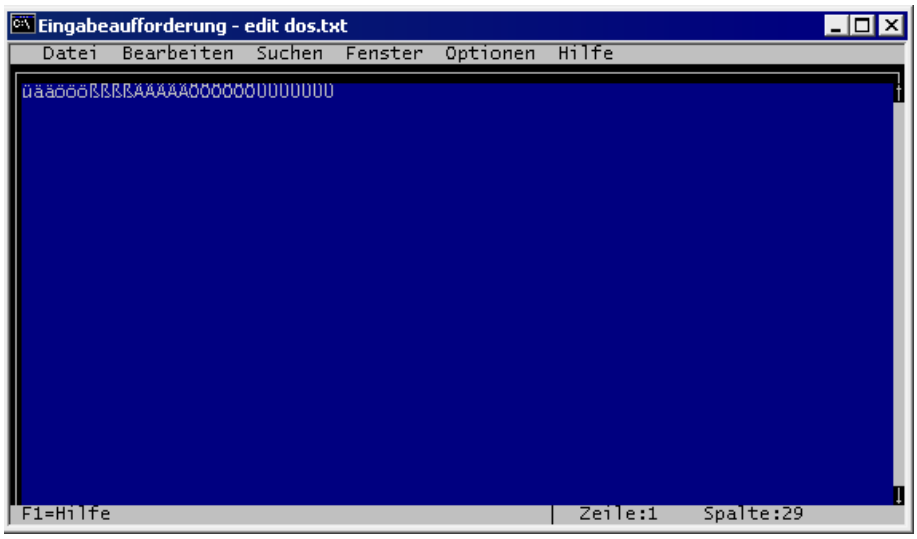


ABBILDUNG 22: ERSTELLUNG DER DATEI „WINDOWS.TXT“ MIT MICROSOFT NOTEPAD

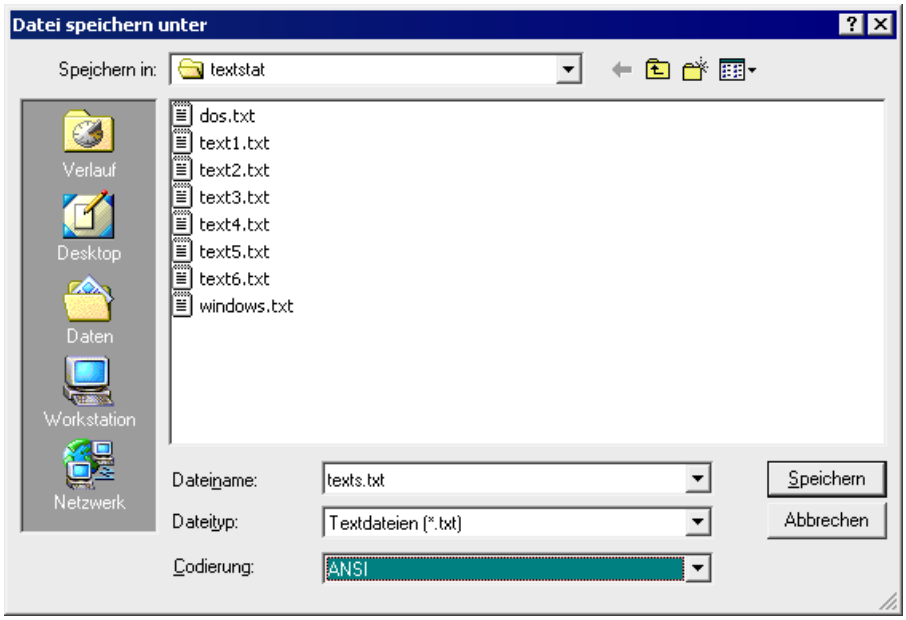


ABBILDUNG 23: AUSWERTUNG DER DATEI „DOS.TXT“ OHNE OPTION „D“

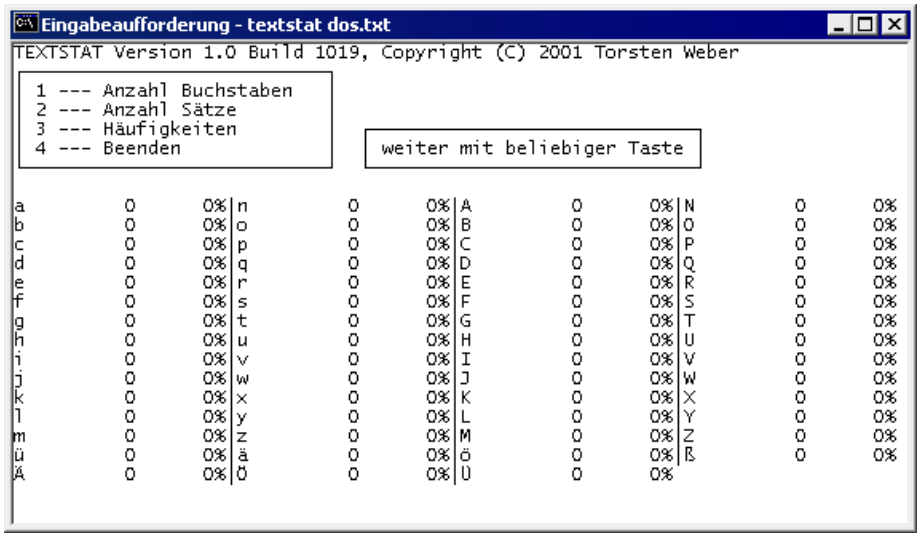


ABBILDUNG 24: AUSWERTUNG DER DATEI ,,DOS.TXT'' MIT OPTION ,,D''

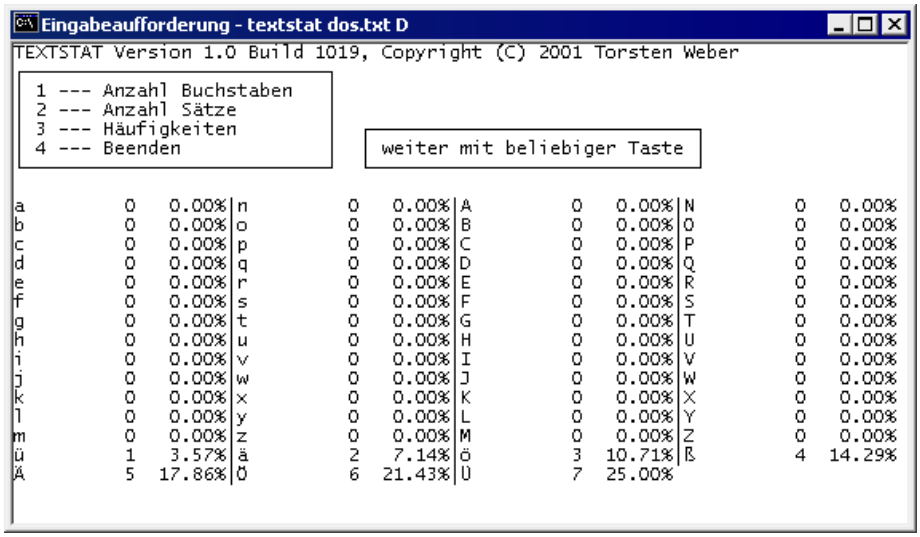


ABBILDUNG 25: AUSWERTUNG DER DATEI ,,WINDOWS.TXT'' OHNE OPTION ,,D''

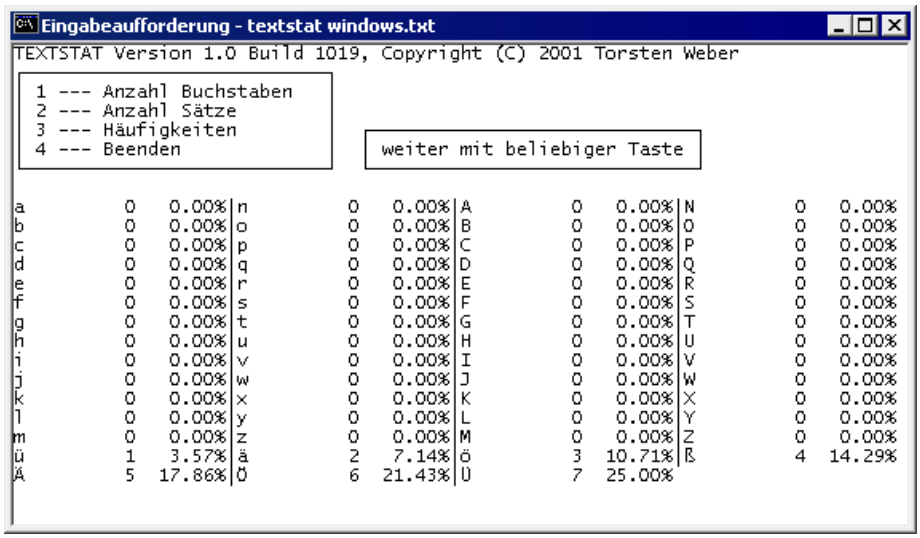
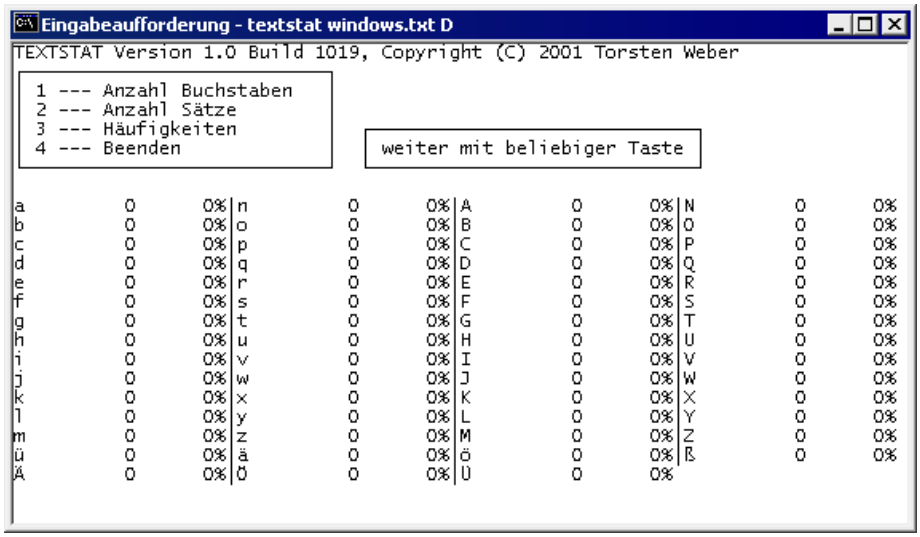


ABBILDUNG 26: AUSWERTUNG DER DATEI ,,WINDOWS.TXT“ MIT OPTION ,,D“



Wie aus den Abbildungen Abbildung 23, Abbildung 24, Abbildung 25 und Abbildung 26 ersichtlich ist, wurde jeweils der Zeichensatz vom Programm[1] richtig beachtet und die entsprechenden Häufigkeiten ausgegeben.

1. Natürlich kann vom Benutzer auch fälschlicherweise der verkehrte Zeichensatz angegeben werden.

6. Stabilitätstests

6.1 Test 11 - Sicherung gegen fehlerhafte Eingaben

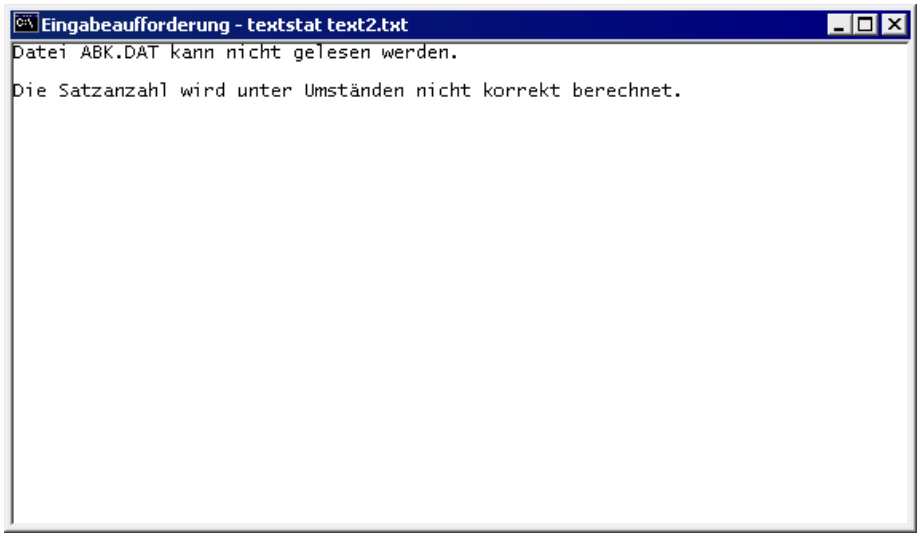
Das Programm ist gegen die fehlerhafte Eingabe gesichert. So ist nur eine Auswahl der vom Programm vorgegebenen Menüpunkte möglich. Bei diesem Test wurden wahllos Tastenanschläge, abgesehen von den Menüpunkten, durchgeführt. Das Programm brach nicht ab, sondern ignorierte diese Eingaben.

Auch Tastenkombinationen - wie z. B. STRG + C[1] - versetzten das Programm nicht in einen undefinierten Zustand und brachen es auch nicht ab.

6.2 Test 12 - Sicherung gegen versehentliches Löschen der Datei ,,abk.dat‘‘

Wurde versehentlicherweise die Datei „abk.dat“ gelöscht, sollte das Programm dem Benutzer eine Fehlermeldung ausgeben, sich aber nicht beenden, da die Buchstabenanzahl / Buchstabenhäufigkeiten auch ohne diese Datei berechnet werden können.

ABBILDUNG 27: PROGRAMMSTART OHNE ,,ABK.DAT‘‘ IM PROGRAMMVERZEICHNIS



Das Programm gab richtigerweise eine Fehlermeldung aus und verarbeitete die angegebene Datei „text2.txt“ korrekt.

1. Abgesehen davon, daß beim Einlesen einer Datei diese Tastenkombination weiterhin verfügbar ist und die Programmausführung weiterhin gestoppt werden kann.

6.3 Test 13 - Sicherung gegen fehlerhafte Angaben beim Programmstart (Parameter)

Sollte der Benutzer beim Programmaufruf eine nicht existierende Datei angeben, darf das Programm nicht in einen undefinierten Zustand geraten und muß eine entsprechende Fehlermeldung ausgeben. Als Parameter wurde „c:\test\test\sinnlos.txt“ angegeben. Weder das Verzeichnis oder die Datei waren vorhanden.

ABBILDUNG 28: ANGABE EINER NICHT EXISTIERENDEN DATEI BEIM PROGRAMMAUFRUF

Das Programm gab richtigerweise dem Benutzer eine Fehlermeldung aus und beendete sich danach.

6.4 Test 14 - Stabilität beim Einlesen einer leeren Datei

Da TEXTSTAT die Häufigkeiten der einzelnen Buchstaben aus

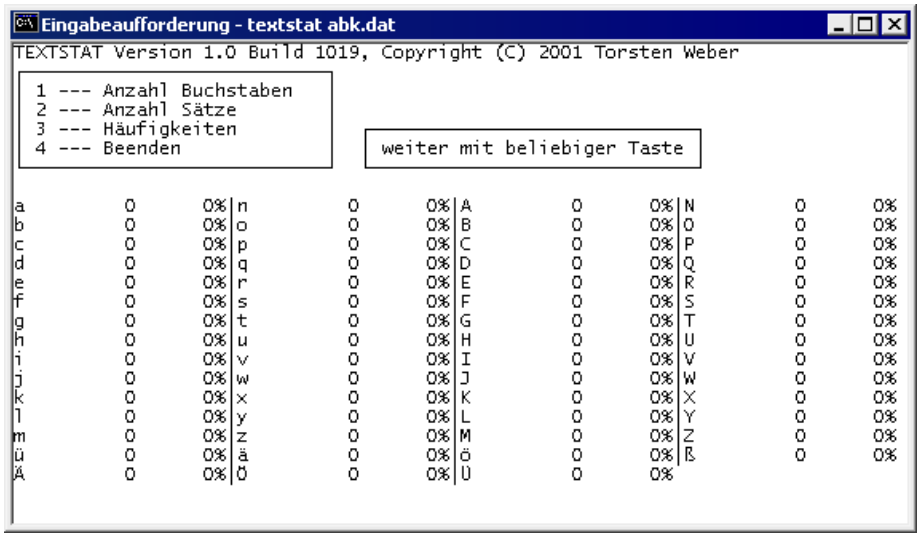
Buchstabe * 100 / (26 * 2 + (7))

berechnet, darf beim Einlesen einer leeren Datei kein Programmabsturz durch

0 / 0

erfolgen. Für diesen Test wurde leere Datei „text6.txt“ angelegt und ausgewertet.

ABBILDUNG 29: AUSWERTUNG MIT TEXTSTAT („TEXT6.TXT“)



Das Programm zeigte kein Fehlverhalten.