Institut für Informatik

Lehrstuhl für Technische Informatik
Prof. Dr. Björn Scheuermann

Digitale Systeme 2019
C-Programmierpraktikum

1 Aufgabenstellung

Leona ist Linguistik-Studentin und Conlangerin. In ihrer Bachelorarbeit beschäftigt sie sich mit einer Menge konstruierter fiktionaler Sprachen, die eine zum Deutschen äquivalente Grammatik besitzen, also mit Sprachen, deren Texte sich aus dem Deutschen durch eine Wort-für-Wort-Übersetzung generieren lassen. Insbesondere möchte Leona für jede der von ihr betrachteten Sprachen eine Übersetzung der Einleitung ihrer Bachelorarbeit mit abdrucken.

Ihre Mission ist es, Leona zu helfen und ein Übersetzungsprogramm zu schreiben, welches einen gegebenen Text in deutscher Sprache mit Hilfe eines ebenfalls gegebenen Wörterbuchs Wort für Wort in eine andere Sprache überträgt.

2 Ein- und Ausgabedaten

Ihr Programm soll ohne jegliche Interaktion und ohne Benutzeroberfläche mit einem einzelnen Aufruf `./loesung demo.wb` von der Kommandozeile aus nutzbar sein. Das erste und einzige Kommandozeilenargument soll den Pfad zur Datei mit dem Wörterbuch enthalten. Der zu übersetzende Text soll von der Standardeingabe (`stdin`) gelesen werden. Jede dieser Eingaben ist in ASCII kodiert, Umlaute sowie „ß“ kommen also nicht vor. Das Wörterbuch enthält pro Zeile ein Wort sowie seine Übersetzung, getrennt durch einen Doppelpunkt. Die Standardeingabe enthält eine beliebige Sequenz druckbarer ASCII-Zeichen sowie Linefeeds. Ihr Programm soll jedes Wort in der Standardeingabe durch seine im Wörterbuch definierte Übersetzung ersetzen. Wörter, die nicht im Wörterbuch definiert werden, sollen an der jeweiligen Stelle im Text in spitze Klammern gesetzt werden. Ein Wort sei hierbei eine Sequenz von Groß- und Kleinbuchstaben, getrennt durch beliebige andere Zeichen. Das Resultat soll Ihr Programm auf die Standardausgabe (`stdout`) schreiben.

Der Einfachheit halber sei Groß- und Kleinschreibung in beiden Sprachen identisch und das Wörterbuch ausschließlich in Kleinbuchstaben definiert. Der erste Buchstabe eines übersetzten Wortes sei genau dann ein Großbuchstabe, wenn der erste Buchstabe im Ursprungstext groß ist; alle übrigen Buchstaben des übersetzten Wortes sind immer klein.

Es folgt ein Beispiel für gültige Ein- und Ausgaben.

demo.wb

```
das:lorem
wissenschaft:commodi
werden:adipisici
ist:dolor
sagt:elit
leben:ipsum
kunst:eiusmod
zu:consectetur
erkannt:incidunt
gelebt:amet
schoenste:tempor
wert:sit
die:sed
```

stdin

```
Das Leben ist wert, gelebt zu werden,
sagt die Kunst, die schoenste Verfuehrerin;
das Leben ist wert, erkannt zu werden,
sagt die Wissenschaft.
```

Friedrich Nietzsche (1844-1900)

stdout

```
Lorem Ipsum dolor sit, amet consectetur adipisici,
elit sed Eiusmod, sed tempor <Verfuehrerin>;
lorem Ipsum dolor sit, incidunt consectetur adipisici,
elit sed Commodi.
```

Und ein weiteres Beispiel:

demo2.wb

```
d:ee
dd:ff
ddd:gg
```

stdin

```
dD1d2ddd3ddDd4d5DD
```

stdout

```
ff1ee2gg3<ddDd>4ee5Ff
```

Wenn Ihr Programm als Wörterbuch eine Eingabe erhält, die nicht der obigen Spezifikation genügt, soll es gar nichts auf die Standardausgabe schreiben. Auf die Standardfehlerausgabe (`stderr`) soll dann eine Fehlermeldung geschrieben werden; als Returncode soll in diesem Fall ein Wert größer als eins zurückgegeben werden.

Insbesondere sollten folgende Fälle als falsche Eingabe (für das Wörterbuch) gewertet werden:

- Ein anderes Zeichen als ein Kleinbuchstabe, Doppelpunkt oder Linefeed tritt auf.
- Eine Zeile enthält nicht genau einen Doppelpunkt.
- Der Doppelpunkt in einer Zeile ist das erste oder das letzte Zeichen.
- Zwei Zeilen beschreiben dasselbe deutsche Wort.

Der Einfachheit halber *dürfen* Sie neben einem einzelnen Linefeed (LF) auch ein carriage return (CR) oder ein „CR LF“ als Zeilenumbruch im Wörterbuch akzeptieren. Wir verzichten darauf, diesbezüglich zu testen.

Der eingegebene Text (stdin) sei genau dann gültig, wenn er ausschließlich die Zeichen 10 (line feed) sowie 32–126(inklusive) enthält. Sollte ein anderes Zeichen im eingegebenen Text auftreten, soll ebenfalls mit einer Fehlermeldung und Returncode größer als eins abgebrochen werden. In diesem Fall darf natürlich bei Abbruch ein Teil der Übersetzung bereits auf die Standardausgabe geschrieben worden sein.

3 Beispieldaten

Für eine gültige Eingabe existiert nur eine gültige Ausgabe. Wir stellen Ihnen im Moodle einige Beispiele für Ein- und Ausgabetripel (`example01.stdin`, `example01.wb`, `example01.stdout`) zur Verfügung, anhand derer Sie selbst überprüfen können, ob Ihr Programm gewisse Eingaben in angemessener Weise bearbeitet. Die Beispiel-Eingaben decken viele — aber nicht alle — Grenzfälle ab, mit denen Ihr Programm zurechtkommen muss.

Für die Überprüfung Ihrer Abgabe werden andere Datensätze zum Einsatz kommen.

Bezüglich falsch formatierter Eingaben oder fehlerhafter Aufrufe müssen Sie selbst kreativ werden. Versuchen Sie, Ihr Programm mit Hilfe bewusst bösartiger Eingaben zum Absturz zu bringen und vergewissern Sie sich, dass für jeden möglichen Programmdurchlauf *niemals* Schutzverletzungen auftreten, nur gültige Lese- und Schreiboperationen auf dem Speicher ausgeführt werden und immer aller vom Heap allozierter Speicher vor Programmende freigegeben wird. *Dies gilt auch für beliebige fehlerhafte Eingaben.*

4 Hinweise zum Algorithmus

Die Aufgabenstellung lässt sich auf folgende Weise lösen:

1. Lesen Sie zunächst das Wörterbuch in eine geeignete Datenstruktur und überprüfen Sie dessen Korrektheit.

2. Lesen Sie die Standardeingabe und übertragen Sie diese Wort für Wort, wobei alle Zeichen, die keine Buchstaben sind, unverändert wiedergegeben werden sollen.

Beachten Sie bei der Auswahl der Datenstrukturen und Ihrer Implementierung einen sorgsamen Umgang mit Speicherplatz. Insbesondere sollte Ihr Programm in der Lage sein, Texte zu übersetzen, die größer sind als der auf dem System zur Verfügung stehende Hauptspeicher. Die Laufzeit für das Übersetzen eines einzelnen Wortes sollte weniger als linear von der Anzahl der Einträge im Wörterbuch abhängen. Gehen Sie sowohl beim Wörterbuch als auch bei der Standardeingabe davon aus, dass Sie die Daten nur einmal sequenziell lesen können, dass also `fseek` und verwandte Operationen auf den Eingaben nicht möglich sind.

5 Zeitplan

Datum	Beschreibung
13.07.2019	Empfohlene Deadline für eine erste Abgabe; bei Abgabe bis zu diesem Termin können wir im Falle von Problemen mit der Abgabe gewährleisten, dass nach einer ersten Rückmeldung noch Zeit zur Nachbesserung bis zur endgültigen Deadline bleibt.
16.08.2019 (23:55 MESZ)	Spätestmögliche Deadline für die Abgabe der Endversion, die alle Kriterien vollständig erfüllen muss.
26.08.2019 – 06.09.2019	In diesem Zeitraum finden die mündlichen Testate in Einzelsitzungen statt. Eine frühere Abnahme ist nach individueller Absprache möglich.

Sie haben *bis zur finalen Deadline* beliebig viele Abgabeversuche.

Danach werden keinerlei weitere Abgaben mehr berücksichtigt!

6 Wichtige Hinweise und weitere Anforderungen

Beachten Sie bei der Erstellung Ihres Programms **unbedingt** folgende Punkte:

- Treffen Sie keine zusätzlichen Annahmen über die Eingabedaten, auch wenn diese plausibel erscheinen. Die Übersetzung von Deutsch in eine fiktionale Sprache soll lediglich der Motivation dienen. In Leonas Welt kann ein Wörterbuch z.B. durchaus deutsche Wörter mit $> 10^6$ Buchstaben enthalten, auch wenn das Ihrer Erfahrung mit natürlichen Sprachen widersprechen mag.
- Halten Sie sich strikt an das oben definierte Ausgabeformat und die Aufrufkonventionen. Wir überprüfen Ihr Programm automatisiert auf die Korrektheit der ausgegebenen Lösungen. *Eine falsch formatierte Ausgabe wird nicht als korrekt anerkannt!* Im Zweifelsfall fragen Sie *rechtzeitig vor Abgabe* nach!
- Verwenden Sie keine Bibliotheken außer der C-Standard-Bibliothek wie sie in der 2011er Ausgabe des ISO C Standard beschrieben ist. Insbesondere Nutzer von Softwareprodukten aus Redmond weisen wir darauf hin, dass Funktionen die einen DromedaryCaseNamen tragen (wie `StrToInt`), sehr wahrscheinlich nicht Teil der C-Standard-Bibliothek sind.

- Ihr Programm soll als eine einzige C-Quelldatei mit dem Namen `loesung.c` in gültigem ISO C11 geschrieben sein. Ihr Programm muss sich mit folgendem einzelnen gcc-7.4-Compileraufruf[1] fehlerfrei übersetzen und linken lassen:
  `gcc -o loesung -O3 -std=c11 -Wall -Werror -DNDEBUG loesung.c`
  Um Fehler bei der Implementierung zu vermeiden, empfiehlt es sich, zusätzlich die Compiler-Flags `-Wextra` und `-Wpedantic` zu verwenden. Verlangt wird dies aber nicht.

- Ihr Programm muss auf der Kommandozeile ohne grafische Oberfläche lauffähig sein und muss genau ein Kommandozeilenargument erfordern, d.h. es soll mit folgendem Aufruf (in `sh`, `bash` oder `zsh`) ausführbar sein:
  `cat demo.in | ./loesung demo.wb`
  Falls Ihr Programm ohne Argumente aufgerufen wird, schreiben Sie eine Fehlermeldung und brechen mit Returncode größer als 1 ab.

- Legen Sie Ihr Programm so aus, dass es mit beliebig großen Eingabedaten umgehen kann. Hierfür ist es unbedingt notwendig, dass Sie dynamische Speicherverwaltung einsetzen. Wir werden Ihr Programm mit *großen* Datensätzen testen!

- Achten Sie darauf, dass Ihr Programm auch auf Systemen mit sehr wenig Speicher keine Schutzverletzung auslöst. Verlassen Sie sich *niemals* darauf, dass ein `malloc` tatsächlich neuen Speicher alloziert. Falls `malloc` fehlschlägt, darf Ihr Programm natürlich abbrechen.

- Ihr abgegebenes Programm wird automatisch auf Speicherlecks überprüft. Stellen Sie sicher, dass es keine Speicherlecks aufweist. Dies können Sie beispielsweise mit dem Werkzeug `valgrind`[2] bewerkstelligen. Geben Sie jeglichen dynamisch zugewiesenen Speicher vor Programmende korrekt wieder frei.

- Stellen Sie sicher, dass Ihr Programm *niemals* eine Schutzverletzung (segmentation fault, segfault) auslöst, egal welche gültige oder ungültige Eingabe es erhält.

- Wir werden alle abgegebenen Lösungen benchmarken. Die ressourcensparsamsten Lösungen werden am Ende des Semesters mit einem kleinen Preis ausgezeichnet. Wir werden hierfür die Lösungen hinsichtlich ihrer Ausführungszeit und ihres Speicherverbrauchs untersuchen.

- Kommentieren Sie Ihren Quelltext in angemessenem Umfang. Im Testat werden Sie Ihren Quelltext detailliert erklären müssen. Dabei können Kommentare helfen.

- Wir werden Feedback zu Ihren Lösungsversuchen ausschließlich als persönliche Moodle-Nachrichten versenden. Sorgen Sie dafür, dass Sie diese Nachrichten rechtzeitig lesen.

6.1 Valgrind

Valgrind ist ein sehr vielseitiges Werkzeug. Im einfachsten Fall testen Sie Ihr Programm mittels
`cat demo | valgrind ./loesung demo.wb 1> /dev/null`

Wenn Ihr Programm keine Fehler in der Speicherverwaltung aufweist, sollten in der Valgrind-ausgabe folgende zwei Zeilen vorkommen:

[1] Den gcc 7.4 finden Sie z.B. auf den Rechnern des Berlin-Pools ([alex|britz|buch|buckow|...].informatik.hu-berlin.de) installiert.

[2] `http://valgrind.org/`

```
==1234== All heap blocks were freed - no leaks are possible
==1234== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```
(Statt „1234“ wird dort in der Regel eine andere Zahl stehen.)

6.2 Randfälle

- Die leere Eingabe ist eine gültige Eingabe. Das gilt sowohl für das Wörterbuch als auch für den Übersetzungstext.
- In ASCII sind die Zeichen 65–90 Großbuchstaben; die Zeichen 97–122 sind die dazugehörigen Kleinbuchstaben. Alle übrigen Zeichen sind keine Buchstaben.
- Bei einem fehlerhaft formatiertem Wörterbuch *muss* Ihr Programm mit einer Fehlermeldung abbrechen und trotzdem allen dynamisch allozierten Speicher wieder freigeben. Selbiges gilt für das Auftreten von anderen Zeichen als 10 oder 32–126 im Text.
- Wenn die Eingabe formal richtig ist, Ihr Programm aber nicht genug Speicher allozieren kann, soll es mit einer entsprechenden Fehlermeldung abbrechen. Beachten Sie aber, dass Ihr Programm alle von uns zur Verfügung gestellten Beispieleingaben mit 64 MiB Speicher bearbeiten können muss. Sie können den Speicher, den Ihr Programm erhalten kann, künstlich mit `ulimit` verknappen. Um den Speicher z.B. auf 1 MiB zu reduzieren, verwenden Sie:
  ```
  ulimit -d 1024
  ```
  Beachten Sie in diesem Zusammenhang auch, dass eventuell ein Limit für die Stackgröße (`ulimit -s`) voreingestellt ist (möglicher Weise 8 MiB). Verwenden Sie Stack-Speicher nicht exzessiv, sondern besser Heap-Speicher.
- Verwenden Sie die folgenden Returncodes:

  0 Wenn die Eingaben formal gültig sind und jedes Wort im Text übersetzt werden kann.

  1 Wenn die Eingaben formal gültig sind, im Text aber Wörter auftreten, die nicht übersetzt werden können.

  >1 In allen echten Fehlerfällen: Wenn die Eingaben fehlerhaft sind, kein Kommandozeilenargument übergeben wurde, die Wörterbuchdatei nicht zum Lesen geöffnet werden kann, nicht genügen Heap-Speicher alloziert werden kann, die Ausgabedatenströme vorzeitig geschlossen wurden oder sonst etwas unvorhersehbares passiert.

Sollten Sie Fragen zur Aufgabenstellung haben, stellen Sie diese bitte im Moodle Forum (`https://moodle.hu-berlin.de/mod/forum/view.php?id=1834859`) und **nicht via E-Mail oder persönlicher Moodle Nachricht**.

Viel Erfolg!