Dr. Johannes Mayer
Axel Blumenstock
Christian Ehrhardt

Abteilung Angewandte Informationsverarbeitung

17. Januar 2006
Blatt 11

# Allgemeine Informatik III / Systemnahe Software I (WS 2005/2006)

**Abgabetermin: 24. Januar 2006**

## ELIZA (10 Punkte)

Vor nunmehr 40 Jahren schrieb Joseph Weizenbaum ein simples Programm namens ELIZA, welches nichts mehr tat, als vom Benutzer eingegebene Sätze in Fragen umzuwandeln und auf gewisse Schlüsselwörter mit passenden Kommentaren zu reagieren.
Weizenbaum war überrascht, wie leicht seine Testpersonen auf die Illusion hereinfielen, es handele sich um einen menschlichen Gesprächspartner. Selbst Psychiater glaubten, mit einem solchen Programm Therapien automatisieren zu können.
Wir wollen ELIZA in einfacher Form nachimplementieren, so dass wir uns (auf Englisch) mit ihr unterhalten können, etwa so:

```
Eliza: Hi. I'm Eliza. Tell me your problems!
You  : Could you help me with my soft exercises?
Eliza: Perhaps you would like to talk about a problem?
You  : What were your compiler options?
Eliza: Why do you ask?
You  : Just interested.
Eliza: What does this mean for you, saying "Just interested"?
You  : I suspect you don't listen.
Eliza: I understand!
You  : No you don't.
Eliza: Don't express you so negative!
```

Gesteuert wird ELIZA von Regeln, die aus einer Textdatei `rules.txt` (bzw. `rules-simple.txt`) eingelesen werden. Ein Beispiel daraus:

```
similar
as like as
!
In which way?
What similarity do you see?
#
```

Eine solche Regel besteht aus (möglicherweise mehreren) Phrasen, eine pro Zeile. Es folgt ein einzelnes „!“, und dann wiederum pro Zeile eine mögliche Antwort. Ein einzelnes „#“ beendet den Antwortblock. Weitere Regeln können folgen.
Die Regeln werden von oben nach unten abgearbeitet. Die erste, bei der eine Phrase in der Benutzereingabe auftaucht, bestimmt ELIZAs Antwort. Dabei wird für jede Regel ein Zähler geführt, der die Antworten dieser Regel reihum auswählt; nach der letzten natürlich wieder die erste. So könnte obige Regel für folgende Konversation sorgen:

```
You  : You're talking as like as a taxi driver.
Eliza: In which way?
You  : Your comments are quite similar.
Eliza: What similarity do you see?
```

### Die Aufgabe

Implementieren Sie ein Programm, welches Regeln aus einer Datei in obigem Format in den Hauptspeicher liest. Dann soll es in einer Schleife an der Standardeingabe Zeilen vom Benutzer entgegennehmen und anhand der Regeln eine passende Antwort ausgeben. Das Programm soll mit Control-D beendet werden können.
Verwenden Sie zur Repräsentation der Zeichenketten hier nun den Typ `stralloc` mitsamt den zugehörigen Funktionen aus der Bibliothek *libowfat* (siehe unten).
Um das Programm übersichtlich zu halten, lohnt sich auch hier eine Aufteilung auf mehrere Quelltextdateien und damit die Erstellung eines Makefiles.

### Zusatzaufgabe (5 Punkte)

Noch ist ELIZA nicht wirklich überzeugend. Das wird deutlich besser, wenn wir sie in die Lage versetzen, Teile der Benutzeraussagen aufzugreifen.
Zu diesem Zweck enthalten etliche Antworten der Regeldatei ein „*“, beispielsweise:

```
Can you
Are you able to
!
Don't you want to * yourself?
You believe I could *?
#
```

Dieses Sternchen soll durch den Text, der auf die gematchte Phrase folgt, ersetzt werden:

```
You  : Can you do my homework?
Eliza: Don't you want to do your homework yourself?
```

Beachten Sie, dass dabei auch gewisse Verben und Pronomina ausgetauscht werden. Zu diesem Zweck steht eine weitere (sicherlich unvollständige) Datei bereit, die in jeder Zeile zwei durch einen Doppelpunkt getrennte Strings enthält, die wechselseitig ausgetauscht werden sollen. Ein Ausschnitt:

```
you:I
your:my
was:were
myself:yourself
```

Stellen Sie dabei durch geeignete Maßnahmen (etwa Markerzeichen) sicher, dass hierbei einmal eingefügte Wörter nicht abermals ersetzt werden.

## Die Bibliothek *libowfat*

Auf den Rechnern der SAI sollte diese Bibliothek unter `/usr/local/diet` installiert sein. Sollten Sie sie (beispielsweise auf Ihrem Heimrechner) nicht vorfinden, können Sie sie aus dem Beispielverzeichnis der Vorlesung herunterladen und wie folgt bauen:

```
$ tar xjf libowfat-0.24.tar.bz2
$ cd libowfat-0.24
$ make
```

Zur Installation (optional) ändern Sie ggf. die Variable `prefix` ganz oben im Makefile auf ein Verzeichnis Ihrer Wahl (also etwa `prefix=/usr/local/diet`) und führen dann als root `make install` aus.
Die Pfade auf das zugehörige Include- bzw. Bibliotheksverzeichnis müssen Sie dann an der gcc-Kommandozeile mit angeben:

```
$ gcc -Wall -I /usr/local/diet/include -L /usr/local/diet/lib \
  prog.c -lowfat
```

Da die *libowfat* statisch gelinkt wird, muss der Verweis auf sie ganz am Ende stehen!
**Beachten Sie, dass diese Bibliothek unter MS Windows/cygwin nicht compiliert.** Sollten Sie nicht umhin kommen, weiterhin unter MS Windows zu programmieren, können Sie durch Editieren des Makefiles weiterkommen. Insbesondere muss die Variable `ALL_OBJS` geändert werden:

```
ALL_OBJS=$(BYTE_OBJS) $(STR_OBJS) $(STRALLOC_OBJS)
```

Durch ein `make libowfat.a` sollte dann eine brauchbare Bibliothek enststehen.

## Tips und Hinweise

- Beachten Sie, dass zur Prüfung, ob eine Regel anwendbar ist, ein bloßes Teilstringmatching nicht ausreicht. Sie müssen auch sicherstellen, dass unmittelbar links und rechts des Teilstrings keine Buchstaben stehen, da sonst beispielsweise „lie“ in „believe“ gefunden würde.
- Sie werden in der Bibliothek möglicherweise nicht alle Funktionen finden, die Sie benötigen. Die für stralloc verfügbaren Funktionen können Sie beispielsweise mit

  ```
  $ ls /usr/local/diet/man/man3/stralloc_*
  ```

  anzeigen. Um die Manpages anzusehen, müssen Sie ggf. die Umgebungsvariable `MANPATH` ergänzen (hier für die bash):

  ```
  $ MANPATH="$MANPATH:/usr/local/diet/man"
  ```

- Für das Einlesen und Ausgeben von strallocs können Sie die Beispiele aus der Vorlesung zu Grunde legen. Weitere Stringfunktionen müssen sie entweder selbst programmieren, oder Sie greifen auf die C-Stringfunktionen zurück. Beachten Sie aber dabei, dass für einen stralloc `sa` der darin enthaltene String `sa.s` normalerweise *nicht* null-terminiert ist. Das müssen Sie durch ein `stralloc_0` erst sicherstellen. Beispiele für hier nützliche Standardfunktionen sind:
  - `char *strstr(char *haystack, char* needle)` sucht nach dem ersten Vorkommen von *needle* in *haystack* und gibt einen Pointer darauf zurück bzw. NULL, falls der Suchstring nicht gefunden wurde.
  - `char *strchr(char *haystack, char c)` sucht in ähnlicher Weise nach dem ersten Vorkommen des Zeichens *c*.
  - `int isalpha(char c)` gibt 1 zurück, genau dann wenn das gegebene Zeichen ein Buchstabe ist.
- Zur Arbeitserleicherung steht auch die Quelltextdatei `arraylist.c` samt Header bereit. Diese implementieren eine arraybasierte Liste mit automatisch angepasster Länge.

**Viel Erfolg!**