ulm university universität
uulm

Inst. für Angew. Informationsverarbeitung
Prof. Dr. Franz Schweiggert
Michaela Weiss
Wolfgang Kaifler

14.12.2010
Blatt 8

Systemnahe Software I (WS 2010/2011)

Abgabetermin: 21.12.2010 (letztes Blatt für die Bachelor Physiker für 2LP)

Aufgabe 1: Theorie (5 Punkte)

- Erklären Sie die Gemeinsamkeiten und Unterschiede von **malloc** und **calloc**.
- Erklären Sie, was **realloc(pnt, sizeof (int))** gefolgt von **realloc(pnt, 0)** macht. Gehen Sie davon aus, dass zu Beginn pnt == NULL gilt.

Aufgabe 2: Dynamischer Vokabeltrainer (4+3+8+8 P.)

In dieser Aufgabe soll der im Übungsblatt 6 implementierte Vokabeltrainer abgeändert werden, so dass er dynamischer ist. Es sollen nun auch Worte länger als 10 Buchstaben unter Berücksichtigung ihres wahren Speicherverbrauchs unterstützt werden und die einmal eingegebenen Vokabeln beim erneuten Programmstart wieder zur Verfügung stehen. Hierzu wird das bisherige Programm *trainerRand.c* in das Einleseprogramm *einlesen.c* und das Trainingsprogramm *training.c* aufgespaltet.

a.) Schreiben Sie das neue Einleseprogramm ***einlesen.c***, welches die Vokabeln zeilenweise in der Form „deutsch englisch“ erwartet und in die Datei *kapitel1.txt* abspeichern. Nutzen Sie hierzu eine **Dateiumlenkung**, welche auch ermöglicht, die bestehende Datei *kapitel1.txt* zu **erweitern**.

Die in Übungsblatt 6 benötigte Angabe, wie viele Vokabeln eingelesen werden sollen, ist nun nicht mehr nötig. Auch die schöne Eingabeaufforderung wollen wir auf Grund der Dateiumlenkung beiseite lassen und erwarten, dass der Benutzer die Vokabeln direkt nach dem Aufruf des Einleseprogramms ohne Meldung immer in der gewünschten Form eingibt.

Die Eingabe kann mit **Strg+d** beendet werden. Nachfolgendes Eingabebeispiel verdeutlicht den Eingabeablauf und das Resultat.

```
thales$einlesen.c ... //TODO: Dateiumlenkung
Hund dog
Katze cat
Maus mouse
(Strg+d)thales$
```

```
thales$ cat kapitel1.txt
Hund dog
Katze cat
Maus mouse
thales$
```

Nun beschäftigen wir uns mit dem Trainingsprogramm ***training.c***, dessen Ablauf wie in Übungsblatt 6, die Vokabelverwaltung jedoch über einen **dynamischen Vektor als lineare Liste** erfolgen soll.

b.) Passen Sie Ihr *vokabel-struct* entsprechend an, dass eine lineare Liste gebaut werden kann. Desweiteren soll das *vokabel-struct* so abgeändert werden, dass *deutsch* und *englisch* char-Pointer sind. Diese Felder müssen vor der Benutzung, d.h. beim Einlesen jeder einzelnden Vokabel, erst gemäß der benötigten Größe alloziert werden. Der Vorteil dieser Speichermöglichkeit ist, dass auch lange Worte aufgenommen werden können und der Speicherplatz passgenau verbraucht wird.

c.) Um die in Teilaufgabe a gespeicherten Vokabeln zu verwenden, müssen diese zuerst aus *kapitel1.txt* eingelesen werden. Vervollständigen Sie hierzu die Funktion *getVocabulary()*, deren Funktionskopf sowie die zum Einlesen aus einer Datei notwendigen Codestücke nachfolgend vorgegeben sind.
Diese Funktion macht aus allen in der Datei *kapitel1.txt* enthaltenen Vokabelnen *vokabel structs*, baut diese zu einer linearen Liste zusammen und gibt einen Zeiger auf das erste Listenelement zurück.
Machen Sie sich zusätzlich eine **globale int-Variable anzahl**, welche von der Funktion auf die Anzahl der eingelesenen Vokabeln gesetzt wird.

```
1  struct vokabel* getVocabulary() {
2   //Datei zum Lesen oeffnen
3   FILE* fd = fopen("kapitel1.txt", "r");
4   if (fd == NULL) {
5    printf("Fehler beim Oeffnen der Datei!\n");
6    exit(1);
7   }
8   ... //TODO
9   //Datei schliessen
10  fclose(fd);
11  return ... //TODO;
12 }
```

Quellcode 1: getVocabulary()

d.) Schreiben Sie die *main-Funktion* des Trainingsprogramms *training.c*, welches zuerst getVocabulary(...) aufruft und danach eine **zufallsgesteuerte** Vokabelabfrage wie in Übungsblatt 6 vornimmt. Diese Abfrage soll nicht über einen Funktionaufruf, sondern direkt in der main erfolgen.

Hinweis:

- Das Einlesen aus einer Datei kann mittels *int fscanf(...)* erfolgen. Hierbei muss als 1. Parameter der Filepointer der zulesenden Datei angegeben werden. Danach folgen die von scanf gewohnten Parameter. Bsp.:

```
13      FILE* fd = fopen("kapitel1.txt", "r"); ...
14      if (fscanf(fd, "%s %s\n", ...) != 2) {...}
```

Quellcode 2: fscanf(...)

- Statt **(*vok).deutsch** können Sie auch kurz **vok->deutsch** schreiben, wenn *vok* ein Pointer auf ein *vokabel-struct* ist und Sie auf das Feld *deutsch* zugreifen möchten.

- Will man Speicher für eine Zeichenkette allozieren, muss man vorher deren Länge kennen und ggf. eingelesene Zeichenketten erst einmal zwischenspeichern. Danach kann die zwischengespeicherte Zeichenkette mittels strcpy(...) auf den nun allozierten Speicherplatz kopiert werden.
  Bei der Speicherallokation ist es wichtig zu beachten, dass ein Byte mehr Platz geholt werden muss, da jede Zeichenkette mit dem Nullbyte abgeschlossen werden muss.
  Bsp.:

```
15      char buffer[256];
16      scanf("%s", buffer);
17      char* word = malloc(strlen(buffer) + 1);
18      if (word != NULL) { //Speicherallokation war erfolgreich
19       strcpy(word, buffer);
20      }
```

Quellcode 3: dynamische Zeichenketten

Viel Erfolg!