ulm university universität
uulm

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

21.12.2010
Blatt 9

Systemnahe Software I (WS 2010/2011)

Abgabetermin: 11.01.2011

Aufgabe 1: Fragen (4 Punkte)

- Erstellen Sie ein Marko `SIG(n)` das welches den Signumswert der übergebenen Zahl `n` zurückliefert:

$$SGN(n) = \begin{cases} -1, & \text{falls } n < 0 \\ 0, & \text{falls } n = 0 \\ 1, & \text{falls } n > 0 \end{cases}$$

  Erläutern Sie anschließend noch, warum ein Aufruf von `SGN(i++)` nicht sinnvoll ist!

- In C wird der zu bersetzende Programmtext durch den sogenannten Präprozessor gefiltert. Es seien die drei Dateien *main.c*, *b.h* und *c.h* gegeben (siehe unten). Geben Sie die Ausgabe des Präprozessors an, wie sie etwa durch das Kommando `gcc -E main.c -D'DEBUG(x)=(puts(x))'` erzeugt wird. (Sie können dabei die durch den Präprozessor normalerweise erzeugten Zusatzzeilen mit Dateinamen und Zeilennummern weglassen.)

main.c

```
Zeile 1
#include "b.h"
#define X
DEBUG("Debugging")
#include "c.h"
HALLO
Letzte Zeile
```

b.h

```
Zeile b1
#define HALLO hallo du
#ifdef X
Zeile b2
#endif
Zeile b3
```

c.h

```
Zeile c1
#ifdef X
HALLO
#endif
#ifdef DEBUG
Debugging
#endif
Zeile c2
```

Aufgabe 2 (19 Punkte)

Nun hat auch beim Weihnachtsmann die neue Technik einzug gehalten. Sein dickes Geschenkebuch wird ab sofort durch eine Textdatei ersetzt.

Um für seine Geschenke auswahl den Überblick der Altersstruktur innerhalb der Beschenkten zu gewinnen, benötigt er ein kleines Programm, welches ihm eine Datei einliest, wahlweise nach Name oder Alter auf- oder absteigend sortiert und wieder (im selben Format) ausgibt.

In dieser Textdatei steht jede Zeile für eine Person. Jede Zeile hat das Format

`<name>:<alter>`

wobei die Felder für Name und Alter keinen Doppelpunkt enthalten.

Das Verhalten des Programms soll durch Kommandozeilenoptionen gesteuert werden können. Jede Option besteht aus '-' vor einem weiteren Zeichen, plus - falls sinnvoll - einem Argument (kann durch Whitespace getrennt sein, muss aber nicht)

Der Benutzer soll per Option nun eine Ein- und Ausgabedatei angeben können sowie. Weiterhin soll es eine Option für den Index einer Spalte geben, nach der Sortiert werden kann und ein Flag, das die Sortierrichtung umkehrt.

Ein Aufruf des fertigen Programms könnte wie folgt aussehen (absteigend Sortiert nach Alter):

```
1  ./a.out -i db.txt -s 1 -r
2 Marion Melone:88
3 Adam Apfel:57
4 Hans Dosenkohl:53
5 Marion Mueller:46
6 Gustav Gurke:30
7 Elvira Erbse:12
```

Bei falscher Verwendung der Optionen soll das Programm seine Benutzung beschreiben und entweder abbrechen oder Defaultwerte verwenden: Beispiel:

```
1  ./a.out db.txt
2 ignoring non-option argument "db.txt"
3 Usage: ./a.out [ -i INFILE ] [ -o OUTFILE ] [ -s {0|1} ] [ -r ]
4   -i FILE ... input file (default stdin)
5   -o FILE ... output file (default stdout)
6   -s N    ... sort by column N
7   -r      ... reverse sort order
```

Evtl. nützliche Bibliotheksfunktionen:

- `void qsort(void *array, int numElements, int sizeOfElement, int (*compareFunction) (const void *, const void *))`

  Sortiert das gegebene Array mit dem Quicksort-Algorithmus. Die Zahl der Elemente im Array und die Elementgröße in Bytes ist als zweiter respektive dritter Parameter anzugeben. Der vierte Parameter ist ein Zeiger auf eine Funktion, die Zeiger auf zwei Elemente erhält und entscheidet, welches davon den höheren Rang haben soll (und damit weiter nach hinten sortiert wird). Die Bedeutung des Rückgabewerts entspricht der von strcmp() : Bei einem Wert größer null ist das erste Element größer als das zweite, bei kleiner null umgekehrt, und bei 0 haben sie gleichen Rang.

Viel Erfolg, Frohe Weihnachten und einen Guten Rutsch ins Neue Jahr!