Fachbereich Mathematik
Prof. Dr. A. Martin
Dipl.-Math. A. Peter
Dipl.-Math. C. Schönberger

TECHNISCHE
UNIVERSITÄT
DARMSTADT

SS 2008
28./29.05.08

4. Tutoriumsblatt zur „Algorithmischen Diskreten Mathematik“

Übung am Computer

Heute wollen wir uns mit dem Einlesen aus Dateien und Schreiben in Dateien beschäftigen. Außerdem lernen Sie wie man Programme schreibt, die man mit Kommando-Zeilen Argumenten aufrufen kann.
Falls Sie noch nicht alle Aufgaben des letzten Blattes bearbeitet haben, schauen Sie sich zunächst diese nochmal an.

Aufgabe C20 (Kommandozeilen-Argumente)
Statt einer Funktion

```
int main (void)
```

kann man auch eine Funktion

```
int main (int argc, char* argv[])
```

definieren.

(a) Implementieren Sie das Programm (`list.c`)

```
#include <stdio.h>

int main (int argc, char* argv[])
{
  int i;

  for (i = 0; i < argc; i++) {
    printf("%d: %s\n", i, argv[i]);
  }

  return 0;
}
```

und führen Sie die folgenden Befehle aus:

```
fb04305: ~ $ ./list
fb04305: ~ $ ./list 1
fb04305: ~ $ ./list -l
fb04305: ~ $ ./list -o Datei -p
fb04305: ~ $ ./list dies sind einige Worte
```

Was ist `argc` und was ist `argv`?

(b) Was für ein Typ ist `argv`?

(c) Was bedeutet das Format „`%s`" in der `printf()`-Anweisung? Sehen Sie z.B. unter `man 3 printf` nach.

Aufgabe C21 (Dateien aneinanderhängen)

In Anhang A des Skriptes ist beschrieben, wie man **aus Dateien liest** und **in Dateien schreibt**. Hier ist nochmal ein kurzes Programmfragment, dass eine Datei in eine andere kopiert.

Hier werden zwei Alternativen vorgestellt: (Characterweises =) Zeichenweises Einlesen mit `fgetc()` und zeilenweises mit `fgets()`. Im folgenden Codefragement sollte jeweils eine Alternative auskommentiert sein.

```
#include <stdio.h>
#include <stdlib.h> /* for abort() */

void copy (char const* source_name, char const* target_name)
{
  FILE* source;
  FILE* target;
  int   c;
  char  buf[MAX_LINELENGHT];

  source = fopen(source_name, "r");
  if (! source) {
    fprintf(stderr, "File '%s' could not be read!\n", source_name);
    abort();
  }

  target = fopen(target_name, "w");
  if (! target) {
    fprintf(stderr, "File '%s' could not be written!\n", target_name);
    abort();
  }

  // Character um Character einlesen und in Zieldatei schreiben
  for (c = fgetc(source); c != EOF; c = fgetc(source)){
    fprintf(target, "%c", c);
  }

  // Alternative: Zeilenweises Lesen und Schreiben
  while (fgets(buf, sizeof(buf), input) != NULL){
    fprintf(target, "%s", buf);
  }

  fclose(target);
  fclose(source);

}
```

Schreiben Sie ein Programm `concat`, dem eine Liste von Dateien auf der Kommandozeile übergeben wird. Das Programm schreibt dann alle Dateien hintereinander in die letzte übergebene Datei. D.h.

```
./concat Datei1 Datei2 Datei3 Alle
```

führt dazu, dass in der Datei `Alle` die Inhalte der Dateien `Datei1`, `Datei2` und `Datei3` hintereinander stehen. Falls die Zieldatei vor dem Aufruf von `concat` bereits existiert, so soll sie überschrieben werden.

Aufgabe C22 (Graphen einlesen)

Schreiben Sie eine Funktion, die eine Datei mit folgender Struktur einliest, und daraus die Arrays `head`, `tail` und `weigth` initialisiert.

In der ersten Zeile der Eingabe, erhalten Sie n und m, die Knoten- bzw. Kantenanzahl des Graphen, mit einem Leerzeichen getrennt. In den m folgenden Zeilen ist jeweils eine Kante, wie folgt, angegeben: Startknoten, Endknoten, und Gewicht jeweils mit einem Leerzeichen getrennt. Sie können annehmen, dass die Knoten mit den Ziffern 1 bis n bezeichnet werden.

Beispiel:

4 3
1 2 0
1 3 0
1 4 0

Die gewünschten Arrays sollen Länge m haben und für jede Kante den Startknoten, Endknoten bzw. das Gewicht enthalten.

Auf der Webseite finden Sie unter Tutorium "Testinstanzen.zip"Beispieldateien.

Hinweis:

Die Funktion `int sscanf(const char *s, const char *Format, ...)` funktioniert ähnlich wie die Ihnen schon bekannte Funktion `scanf(const char *Format, ...)`. Der Unterschied besteht darin, dass sie nicht vom Standard-Input liest, sondern aus dem String, auf den ihr erstes Argument zeigt. Ähnlich gibt es auch eine Funktion
`fscanf(FILE *Datei, const char *Format, ...)` die aus der Datei, auf die `Datei` zeigt, liest.

Aufgabe C23 (**Challenge: Kürzeste Wege**)

Implementieren Sie einen Kürzeste-Wege-Algorithmus für nicht-negative (ganzzahlige) Kantengewichte.
Ihr Programm soll den Graphen aus einer Datei mit in Aufgabe C22 beschriebenem Format einlesen, und die Länge der kürzesten Wege zwischen Knoten 1 und allen anderen Knoten ausgeben. Der Dateiname soll als Kommando-Zeilen-Argument übergeben werden. Sie können annehmen, dass der Graph immer zusammenhängend ist.

Damit wir die Ergebnisse überprüfen können, soll die Länge jedes Weges (ohne weitere Ausgabe) in einer eigenen Zeile ausgegeben werden. Die Ausgabe soll in Reihenfolge der Knotennummern erfolgen.

Beispiel:
Die Distanz von Knoten 1 zu Knoten 2 ist 42, zu Knoten 3 52 und zu Knoten 4 34. In diesem Fall ist Ausgabe einfach:
42
52
34

(a) Starten Sie mit einer Implementieren des Dijstra-Algorithmus aus dem Skript.

(b) Verbessern Sie Ihre Implementation. Unternehmen Sie dazu gerne auch eine Literaturrecherche.

Testen Sie Ihr Programm an den auf der Webseite zum Download bereit gestellten Instanzen mit 4, 20, 50, 100 und 200 Knoten. Zu diesen ist auch jeweils die richtige Lösung zum Vergleich bereitgestellt. Testen Sie ihr Programm auch an den größeren bereitgestellten Instanzen. Für die beste Lösung gibt es eine Kiste Bier oder Äquivalentes zu **gewinnen**!

Sie können diese Aufgabe gerne in Zweier-Teams bearbeiten. Wenn Sie am Wettbewerb teilnehmen möchten, schicken Sie Ihren (auf den Mathebau-Linux-Rechnern) lauffähigen Code bis zum Dienstag 24. Juni an schoenberger@mathematik.tu-darmstadt.de. Wir wollen dann mit Ihnen einen Termin am 26. oder 27. Juni vereinbaren an dem wir gemeinsam das Programm testen. Wunschtermine können Sie gerne mitangeben. Die Siegerehrung soll dann in der letzten Vorlesung am 1. Juli stattfinden.

Bei Fragen und Problemen können Sie gerne in unserer Sprechstunde freitags, 13:00 bis 14:00 Uhr vorbeikommen. Sie finden uns in Raum 315. Wir können diese Sprechstunde dann auch gerne in den Rechnerraum verlegen.