6.2.A Extras/Lesen aus Dateien – Arbeitsblatt

Aufgabe 1: Ziffernzeichen in einer Datei zählen

Schreiben Sie ein Programm `digitcount.c`, welches die Anzahl der Ziffernzeichen in einer Datei zählt! Folgendes soll in diesem Programm passieren:

- Eingabe des Pfadnamens einer Datei von der Tastatur
- Lesen des Dateiinhaltes
- bei gleichzeitigem Zählen der Ziffernzeichen in dieser Datei.
- Ausgabe des Ergebnisses

Aufgabe 2: Worte einer Datei zählen

Das Programm `wordcount.c` soll die Worte einer Datei zählenAls Wort dient jede Ansammlung von Zeichen, die durch ein oder mehrere Leerzeichen, Tabulatoren oder Zeilenumbrüche getrennt ist.

a) Versuchen Sie zuerst, die Worte in einer Zeichenkette zu zählen. Dabei kann das folgende Zustandsdiagramm in Abbildung 1 helfen.

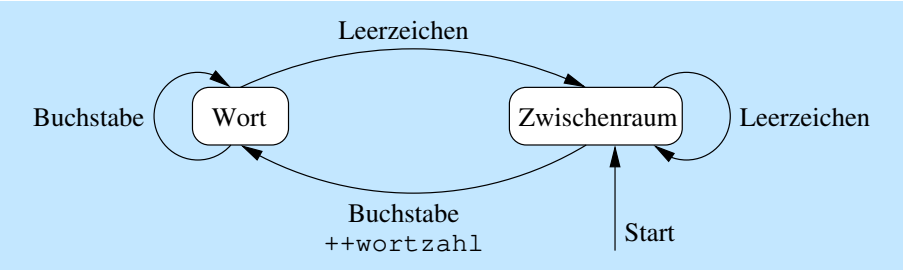

Abbildung 1: Zustandsdiagramm zu `wordcount.c`

b) Jetzt können Sie mit `getchar()` die Zeichen von der Tastatur einlesen und in diesem Datenstrom die Worte zählen.

c) Im letzten Schritt lesen Sie mit `fgetc()` die Zeichen aus einer Datei ein.

Aufgabe 3: Einlesen von Zahlen

Das Programm `readnumbers.c` soll aus einer Datei (Pfadname wird per Tastatur eingegeben) Zahlen einlesen (Kommazahlen sollen erlaubt sein). Die kleinste Zahl, die größte Zahl und der arithmetische Mittelwert sollen berechnet und auf dem Bildschirm ausgegeben werden.

Aufgabe 4: Bildbearbeitung mit PNM-Dateien

Es soll ein Programm geschrieben werden, mit dem Bild-Dateien im PGM-Format gelesen und geschrieben werden können.

a) Was ist der Unterschied zwischen Pixel- und Vektorformaten?

b) Informieren Sie sich über Dateien im PGM-Format (Stichworte: NetPBM, Portable Any-Map)! Hier soll das Format P2 verwendet werden.

c) Mit welchem Programm kann man PGM-Dateien anzeigen (mehrere Nennungen möglich)?

d) Erstellen Sie im Speicher ein mittelgraues Bild der Größe 200x100 Pixel (Breite mal Höhe) und schreiben Sie es in eine Datei! Zeigen Sie die Datei auf dem Bildschirm an! (Funktion: `schreiben()`)

e) Erstellen Sie nun die Funktion `lesen()`, mit der Sie eine Datei einlesen können! Testen Sie die Funktion, indem Sie eine Datei einlesen und sofort wieder (unter anderem Namen) schreiben. Vergleichen Sie dann die beiden Dateien!

f) Nun können Sie die Funktion `heller()` erstellen, die Ihr Bild aufhellen soll. Zusätzlich sind weitere modifizier-Funktionen möglich.

Hier ist ein Beispielcode, an den Sie bei Ihrer Arbeit anknüpfen können:

```
1  #define WMAX 800
2  #define HMAX 1200
3
4  int lesen(const char *dateiname, unsigned char* buf, size_t
5            bufsize, unsigned int *pw, unsigned int *ph);
6  int schreiben(const char *dateiname, const unsigned char* buf,
7                unsigned int w, unsigned int h);
8  void heller(unsigned char *buf, unsigned int w, unsigned int h);
9
10 int main(void)
11 {
12    int rc;
13    unsigned int w, h;              /* Breite, Hoehe */
14    unsigned char buf[WMAX*HMAX]; /* Speicher fuer das Bild */
15
16    rc=lesen("windmuehle.pgm", buf, sizeof(buf), &w, &h);
17    if(rc)
18    {
19       printf("lesen(): %d!\n", rc);
20       return -1;
21    }
22    heller(buf, w, h);
23 /*    weicher(buf, w, h);    */
24 /*    koernung(buf, w, h);   */
25    rc=schreiben("windmuehle_kopie.pgm", buf, w, h);
26    if(rc)
27    {
28       printf("schreiben(): %d!\n", rc);
29       return -1;
30    }
31    return 0;
32 }
```

Hinweis: Mit dem Programm `pnmtoplainpnm` kann man ein Bild in gepacktem Format (Binärdatei) in ein Bild in ausführlichem Format (Textdatei) konvertieren (also P6 nach P3, P5 nach P2 und P4 nach P1).