Praktikum 9 Textdateien

Schreiben Sie ein C-Programm woerter.c, das aus einer Datei unter Benutzung der Funktion fgets Textzeilen einliest, und diese unter Verwendung der Funktion **puts** auf der Standardausgabe ausgibt. Der Dateiname soll über die Kommandozeile eingegeben werden. Die Textzeilen sollen maximal 128 Zeichen lang sein. Von der Shell lässt sich das Programm aufrufen unter

```
woerter irrtum.txt | more
```

Dabei wird die Eingabe aus der Datei irrtum.txt gelesen. Über die Pipe | more wird jeweils nach einer Bildschirmseite angehalten. Erweitern Sie dieses Programm, um einen Mechanismus, der die Menge der Wörter gleicher Längen (der Längen 1 bis 10) zählt und folgende Ausgabe erzeugt:

```
Woerter der Laenge:
 1:   3 gefunden
 2:  12 gefunden
 3:  46 gefunden
. . .
10:   4 gefunden
11: sonstige Woerter
```

Es gebe keine Wörter, die über eine Zeile hinausgehen oder abgetrennt sind.

Zur Implementierung ist es günstig, ein Array von 11 Int-werten zu definieren und mit Nullen zu initialisieren. Hat man die Länge eines Wortes bestimmt, so verwendet man die Länge, so sie nicht größer als 10 ist, als Index in dieses Array und incrementiert den darin enthaltenen Zähler. Im Eintrag [0] kann man die restlichen (längeren) Wörter zählen.
Zur Bestimmung der Wortlängen kann die Verwendung der Funktion strtok hilfreich sein. Suchen Sie nach Informationen zur Verwendung von strtok.
Beispieldateien sind in Opal zu finden.

Zusatz:

Optimieren Sie das Programm, in dem Sie zunächst das längste Wort finden. Danach können Sie dann das Array für die Counter zu den einzelnen Wortlängen lokal anlegen. Damit entfällt dann die „Zeile sonstige …“

Praktikum 10 Funktionspointer

Funktionspointer bieten eine hervorragende Flexibilität in C-Programmen. Vor allem bei der Programmierung graphischer Oberflächen mit verschiedenen C-Toolkits treten sie in Form sogenannter Call-back-Funktionen häufig auf.
Eine andere Anwendung ergibt sich in Verbindung mit der Funktion qsort aus der Standardlibrary. Diese Funktion dient dem Sortieren beliebiger Daten, vorausgesetzt, die einzelnen Daten sind alle von der selben Größe. Da die Funktion qsort die Daten nicht kennt, die sie sortieren soll, muss man ihr ein paar Informationen geben.
1. Die Adresse der zu sortierenden Daten.
2. Die Anzahl der zu sortierenden Daten(-sätze)
3. Die Größe eines jeden einzelnen Datensatzes oder Datums
4. Eine Funktion zum Vergleich zweier Daten(-sätze)
Die Funktion hat somit folgenden Prototyp:

```
void qsort(void *base,
           size_t nmemb,
           size_t size,
           int(*compar)(const void *, const void *));
```

Der letzte Parameter ist ein Pointer auf eine Funktion mit der 2 Daten aus dem zu sortierenden Bestand verglichen werden. Ihr Returnwert sollte dem von strcmp oder memcmp entsprechen, also kleiner 0, gleich 0 oder größer 0 sein, je nach Ergebnis des Vergleiches. Diese Funktion bekommt 2 const void* Pointer übergeben, d.h. innerhalb der Funktion müssen diese Pointer auf ihren tatsächlichen Typ, den Typ der zu vergleichenden Daten mittels cast typgewandelt werden.
Eine bespielhafte Vergleichsfunktion zum Vergleichen zweier Währungen (Praktikum Strukturen) könnte folgendermaßen aussehen:

```
int cmpWaehrungsbez(const void* p1, const void* p2)
{
  return strcmp (((tWrg*)p1)->Waehrungsbez, ((tWrg*)p2)->Waehrungsbez);
}
```

1. Modifizieren Sie Ihre Lösung zu Praktikum 8 so, dass Ihr Programm die Daten unter Verwendung von qsort nach Land, Landeskürzel oder Währungsbezeichnung ausgeben kann.

2. Schreiben Sie eine universelle Sortierfunktion mysort, die genau das selbe macht, wie die Funktion qsort, also die Daten, die in einem Vektor gespeichert sind, aufsteigend/fallend sortieren kann.Das Sortierverfahren können Sie frei wählen. Testen Sie die Funktion, in dem Sie die Daten waehrung.dat aus Praktikum 8 wahlweise nach Land, Landeskürzel oder Währungsbezeichnung unter Verwendung Ihrer Funktion sortiert ausgeben.
Zusatzaufgabe:

Vereinbaren Sie einen Pointervektor und lassen Sie qsort , bzw. mysort nur die Pointer sortieren. Die Lösung bedarf einer Anpassung der zu übergebenden Vergleichsfunktion.

A. Beck