Übung vom 08.07.98 bis 14.07.98

Aufgabe 30:

Entwickeln Sie in den Übungen die nachfolgenden, zum Sortieren von Wörtern notwendigen Funktionen. Die zu bearbeitenden Worte sollen in einem 2-dimensionalen Feld (kein globales Feld!) **char words [MAX_STRINGS][MAX_WORDLENGTH]** abgelegt werden. Desweiteren existiere ein Feld von Zeigern **char *all_words[MAX_STRINGS]**, welches als Liste der eingelesenen Worte dienen soll. Die Anzahl der eingelesenen Worte sei in der Variablen **no_of_words** abgelegt.

a) **int string_cmp(char *s1, char *s2):**
Diese Funktion vergleicht die beiden übergebenen Zeichenketten und liefert als Ergebnis:
-1 falls die Zeichenkette s1 lexikographisch kleiner ist als s2,
0 falls beide Zeichenketten identisch sind
1 falls die Zeichenkette s1 lexikographisch größer ist als s2

b) **char *next_word(char *word):**
Diese Funktion liest das nächste Wort (= Folge von alphanumerischen Zeichen inklusive des Zeichens '-') von der Standardeingabe. Beim Aufruf soll dieser Funktion ein Zeiger auf die nächste freie Zeile des Feldes **words** übergeben werden. Ist man am Ende der Eingabe angelangt (d.h. das aktuelle Zeichen ist EOF), so liefert die Funktion den Wert NULL zurück. Andernfalls wird ein Zeiger auf das eingelesene Wort zurückgeliefert. Worte die länger als **MAX_WORDLENGTH** sind, sollen abgeschnitten werden.

c) **int insert_word(char *string, char *all_words[], int no_of_words):**
Diese Funktion fügt einen Zeiger auf ein eingelesenes Wort in die Wortliste **all_words** ein. Dabei soll zunächst überprüft werden, ob das Wort bereits in der Liste vorhanden ist. Zu diesem Zweck sollte die Funktion **string_cmp** verwendet werden. Ist dies der Fall, so soll **insert_word** den Wert 0 zurückgeben. Andernfalls soll der Wert 1 zurückgeliefert werden.

Entwickeln Sie dabei nicht nur eine Lösung als C-Programm, sondern erstellen Sie vorher jeweils ein Struktogramm.

Aufgabe 31:

Führen Sie einen Trace des untenstehenden Programmes durch. Geben Sie den Inhalt der Variablen nach jeder erfolgten Änderung in einer Tabelle an!

```
int k[] = {1,0,1,0};
void f(int *k)
{
    int m=*k-2;
    do {
        m = k[1];
        *(k+1) += *k;
        k++;
    }while(m>0);
```



```
        (*k)++;
    }

    int main()
    {
        int *m = k;
        f(m);
        m++;
        f(m);
        return *m;
    }
```

Aufgabe 32:(Abgabe bis Mi. 22.07.98, 12 Uhr)

Entwickeln Sie ein Programm, das einen Text zeichenweise von der Standardeingabe einliest. Dabei soll es, je nach den in der Kommandozeile übergebenen Argumenten, entweder den Text in Wörter zerlegen und diese alphabetisch sortiert ausgeben oder die Anzahl der Zeichen/Wörter/Zeilen des Textes zählen.

a) Jedes mögliche Argument (-s für sortieren, -c für Anzahl der Zeichen, -w für Anzahl der Wörter und -l für Anzahl der Zeilen) soll in der Main-Funktion durch eine Variable repräsentiert werden, die auf "wahr" gesetzt wird, wenn das ihr entsprechende Argument in der Kommandozeile vorhanden ist.
Diese Aufgabe soll von einer Funktion erledigt werden, d.h. sie muß ggf. die o.g. Variablen ändern können. Werden unbekannte Argumente auf der Kommandozeile übergeben, so soll die Funktion eine Fehlermeldung ausgeben und das Programm beenden (siehe "man exit"). Das gleiche soll passieren, wenn -s zusammen mit einer der anderen Optionen verwendet wird. Dagegen ist die gleichzeitige Verwendung von -c, -w und -l erlaubt. Falls das Programm ohne Argumente aufgerufen wird soll es sich so verhalten als wäre -w angegeben.

b) Die Bearbeitung des eingegebenen Textes soll bei den Argumenten -c, -w und -l (oder bei einer Kombination davon) von **einer** Funktion vorgenommen werden. Diese soll den Text zeichenweise einlesen (siehe "man getchar") und dabei mitzählen, wieviele Zeichen/Wörter/Zeilen vorhanden sind. Zur Bestimmung der Wortgrenzen können sie die vorhandenen C-Funktionen verwenden (siehe "man isalnum", "man isspace", usw.).

c) Eine andere Funktion soll die Verarbeitung des Textes erledigen, wenn die Option -s verwendet wird. In dieser Funktion müssen die in Aufgabe 30 erwähnten Felder definiert werden, die dann mit den ebenfalls vorgestellten Funktionen next_word() und insert_word() gefüllt werden können. Achten Sie hierbei darauf, daß ein Wort nur einmal im Feld abgespeichert wird. Nun soll noch in alphanumerisch aufsteigender Reihenfolge (d.h. so wie von string_cmp() in Aufgabe 30 geordnet) sortiert (siehe Teilaufgabe d) und die Wörter in der neuen Reihenfolge augegeben werden.

d) Sie benötigen zwei weitere Funktionen zum sortieren des Feldes all_words: Eine Funktion swap() zum Vertauschen von zwei Elementen des Feldes, sowie eine Funktion string_sort(), die die Feldelemente (bzw. die Strings, worauf die Elemente zeigen) vergleicht und ggf. vertauscht (man erinnere sich an den Bubble-Sort...).

e) Geben Sie zu der im Aufgabenteil a) programmierten Funktion ein Struktogramm ab.

Sie finden unter /proj/i4cing/CountAndSort/ ein Gerüst für Ihr Programm, das u.a. auch die in Aufgabe 30 vorgestellten Funktionen enthält (es ist alles auch im WWW verfügbar unter http://www4.informatik.uni-erlangen.de/Lehre/SS98/V_C/Uebung/). Dort liegt außerdem ein Text, den Sie zum Testen Ihres Programms verwenden können. Wenn das Programm z.B. "cas" heißt könnte ein Testlauf z.B. so aussehen:

```
> cas -l -c < A32.txt
Zeichen: 3115
Zeilen: 65
```

<u>Noch einige Hinweise zur Klausur:</u>

- Datum / Ort: Do. 30.07.1998 um 8 Uhr in der Mensa
- Es sind keine Hilfsmittel erlaubt
- Zugelassen zur Klausur wird, wer mindestens 5 Aufgaben korrekt bearbeitet hat. Eine Liste mit dem jeweiligen Stand der Übungsteilnehmer wird voraussichtlich am Mi. 15.07.1998 über dem Abgabekasten ausgehängt.
- Einige der in den letzten Jahren gestellten Klausuren liegen unter /proj/i4cing/Klausuren/ oder im WWW: http://www4.informatik.uni-erlangen.de/Lehre/SS98/V_C/Klausuren/