

**Wichtig:** Lesen Sie auch den Teil **“Hinweise zur Aufgabe”** auf diesem Blatt; Spezifikationen in diesem Teil sind ebenfalls einzuhalten!

## Aufgabe 2: wsort (10.0 Punkte)

### Standard-Eingabe sortieren

Schreiben Sie ein Programm `wsort`, welches eine Liste von Wörtern vom Standard-Eingabekanal (`stdin`) einliest, diese Liste alphabetisch sortiert und die sortierte Liste auf dem Standard-Ausgabekanal (`stdout`) ausgibt.

Sowohl beim Einlesen als auch beim Ausgeben der Wörter steht jedes Wort in einer eigenen Zeile. Ein Wort umfasst **alle** Zeichen einer Zeile. Zeilen sind durch ein Zeilenumbruch-Zeichen (`\n`) voneinander getrennt, das selbst nicht Teil des Wortes ist. Jede Zeile endet mit einem Zeilenumbruch – lediglich die letzte Zeile muss nicht zwangsläufig ein `\n`-Zeichen enthalten.

Wörter, die eine maximale Länge von 100 Zeichen überschreiten, werden mit einer entsprechenden Fehlermeldung ignoriert. Leere Zeilen sind ohne Fehlermeldung zu ignorieren.

Im zip-Archiv finden Sie Beispiel-Eingabedateien (`wlist*`) sowie eine Vergleichsimplementierung (`wsort.bsteam`), mit der Sie die Wortlisten sortieren und die Ausgabe jeweils mittels **diff(1)** mit der Ausgabe Ihres eigenen `wsort`-Programms vergleichen können.

*Hinweis:* Falls beim Versuch `./wsort.bsteam` auszuführen ein Fehler *Permission denied* auftritt, dann kann es helfen die Berechtigung executable der `wsort.bsteam` zu setzen: `chmod +x wsort.bsteam`

**Selbst wenn alle Beispiel-Eingabedateien von Ihrer Lösung korrekt sortiert werden, können dennoch Fehler in Ihrer Implementierung enthalten sein. Verlassen Sie sich daher beim Testen nicht alleine auf die Beispiel-Eingabedateien.**

Mit Hilfe von **malloc(3)** und **realloc(3)** können Sie dynamisch Speicher an- und nachfordern, um die benötigten Datenstrukturen anzulegen bzw. zu erweitern.

### Hinweise zur Aufgabe:

- Hilfreiche *Manual-Pages*: `feof(3)`, `ferror(3)`, `fflush(3)`, `fgetc(3)`, `fgets(3)`, `fputs(3)`, `getchar(3)`, `malloc(3)`, `qsort(3)`, `realloc(3)`, `strchr(3)`, `strcmp(3)`, `strlen(3)`
- Alle zur Ein-/Ausgabe genutzten Funktionen (mit Ausnahme von **`feof(3)`**, **`ferror(3)`**) setzen die `errno`, auch wenn dies ggf. nicht aus der *Manual-Pages* hervorgeht.
- Sämtliche Fehlermeldungen sollen auf dem Standardfehlerkanal (`stderr`) ausgegeben werden. Auf die Standardausgabe (`stdout`) soll **ausschließlich die sortierte Wortliste ausgegeben werden.**
- Achten Sie auf korrekte Fehlerbehandlung bei Funktionen zur Eingabe **und** Ausgabe! Auch Ausgaben auf `stdout` können fehlschlagen, wenn die Ausgabe in eine Datei umgeleitet wird.
- **Strg-D sendet EOF an die Anwendung und beendet somit die Eingabe.**
- Beachten Sie, dass bei der Korrektur `valgrind` zum Einsatz kommen wird. **Testen Sie daher Ihre Lösung mit `valgrind`** auf eventuelle Speicherzugriffsfehler und beseitigen Sie diese.
- Ihr Programm muss mit den folgenden Compiler Flags übersetzen:  
`-std=c11 -pedantic -Wall -Werror -D_XOPEN_SOURCE=700`  
 Diese Flags werden zur Bewertung herangezogen.

### Hinweise zur Abgabe:

Bearbeitung:	Zweiergruppen
Abzugebende Dateien:	<code>wsort.c</code> (10 Punkte)
Abgabezeitpunkte nach Tafelübungsgruppe:	T01-T02: 07.05.2023, 17:30 Uhr T03-T06: 10.05.2023, 17:30 Uhr T07-T09: 11.05.2023, 17:30 Uhr