Fakultät IV - Elektrotechnik und Informatik
Einführung in die Programmierung WS 2017/18
Feldmann / Foucard / Lichtblau / Streibelt / Krüger / Enghardt / Palmer

Aufgabenblatt 4

letzte Aktualisierung: 01. Dezember, 11:26 Uhr

(d16ba531ef149a7db214ca5815b8e44785d91339)

Ausgabe: *Freitag, 01.12.2017*
Abgabe: *spätestens Mittwoch, 13.12.2017, 18:00*

Thema: *Listen & Datenstrukturen*

Abgabemodalitäten

- Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des eecsIT mittels `gcc -std=c99 -Wall` kompilieren.
- Für die Hausaufgaben im Semester gibt es ein eigenes Subversion-Repository (SVN). Bitte die Hinweise auf Blatt 1 dazu beachten!
- Die Abgaben für Blatt 1 bis 4 erfolgen als Einzelabgaben. Ab Blatt 5 erfolgen Abgaben in festen Gruppen à 2 Personen.
- Die Gruppen werden vor Ausgabe von Blatt 5 gebildet. Solltet ihr keine Gruppe gebildet haben, werdet ihr einer Gruppe zugewiesen.
- Alle Abgaben müssen an der dafür vorgesehenen Stelle die Namen aller Gruppenmitglieder bzw. bei Einzelabgaben des Autors/der Autorin enthalten!
- Die Abgabe erfolgt ausschließlich über unser SVN im Abgaben-Ordner. Die finale Abgabe
  - für Gruppenabgaben erfolgt im Unterordner
    `Tutorien/t<xx>/Gruppen/g<xx>/Abgaben/Blatt<xx>/`
  - für Einzelabgaben erfolgt im Unterordner
    `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Abgaben/Blatt<xx>/`

  WICHTIG: Die Abgabeordner werden immer von uns erstellt, sonst kommt es zu Konflikten! Benutzt zum Aktualisieren `svn up` im obersten Verzeichnis des Repositories!
- Benutzt für alle Abgaben die in der jeweiligen Aufgabe angegebenen Dateinamen (die von uns vorgegebenen Dateien haben das Wort ”vorgabe” im Dateinamen, du musst die Datei mit deiner Lösung also entsprechend umbenennen.)
- Gib bei den Programmieraufgaben den C-Quellcode ab und achte darauf, dass die Datei mit .c endet
- Bei Textaufgaben sind, wenn nicht anders angegeben, .txt Dateien zugelassen, die als Plaintext-Datei zu speichern sind. (Keine Word-Dateien umbenennen, etc.!)

Hinweise zur Gruppenarbeit, beginnend mit Aufgabenblatt 5

Ab dem **fünften** Blatt werden die Aufgaben grundsätzlich in festen Gruppen, die aus zwei Studierenden bestehen, gemeinsam bearbeitet und eingereicht.

Dazu müsst Ihr selbstständig Gruppen bilden! Solltest Du keine/n GruppenpartnerIn finden, werden die Gruppen zufällig zugewiesen.

Die Gruppenbildung wird über das euch bekannte Webinterface, Osiris, organisiert.

Besuche dazu folgende URL:
https://teaching.inet.tu-berlin.de/services/osiris-wise1718/topple/groups

Hier kannst Du über den angezeigten grünen Button bis zu 2 Gruppenvorschläge anlegen und jeweils eine andere Person einladen, mit Dir eine Gruppe zu bilden. Dazu benötigst Du den Benutzernamen der anderen Person. Gebe diesen in der Maske ein, so wie Du Dein Login beim svn-checkout angibst. Der Benutzername/das Login steht auch fettgedruckt oben rechts auf jeder Osiris-Seite hinter ’angemeldet als’. Die eingeladene Person bekommt dann in Osiris Deinen Namen und die Nachricht, die Du angegeben hast angezeigt.

Wenn Du eingeladen wirst, eine Gruppe zu bilden, siehst Du auf der oben genannten Seite eine Liste der eingegangenen Einladungen und kannst diese annehmen oder ablehnen und ebenfalls eine Nachricht als Antwort angeben.

Sobald Du eine Einladung angenommen hast oder eine Person die Du eingeladen hast Deine Einladung annimmt, wird die Gruppe automatisch angelegt.
Hinweis: Der Gruppenordner im SVN-Repository wird jedoch nicht vor Veröffentlichung von Blatt 5 erscheinen.

Es werden keine Benachrichtigungen per Email versandt, Du musst also selbst auf der Seite nachsehen, ob Du eingeladen wurdest.

Weitere Details findest Du auf Osiris während der Benutzung der Funktion.

Wichtig: Beachte bitte, dass aus organisatorischen Gründen nur Gruppen innerhalb eines Tutoriums gebildet werden können. Falls Du bereits eine/n GruppenpartnerIn aus einem anderen Tutorium hast, liegt es im Ermessen der TutorInnen, ob Du das Tutorium tauschen kannst. Sprich dazu bitte die Person an, in deren Tutorium Du wechseln willst und klärt mit dem Tutor/der Tutorin, ob ein Wechsel möglich ist. Den Wechsel führen dann die Tutoren durch.

1. Aufgabe: Eine Büchersammlung als Liste (2 Punkte) *[Einzelabgabe]*

Ziel der Aufgabe ist es, Daten zu Büchern aus einer Datei in eine einfach verkettete Liste zu laden und diese mithilfe einer vorgegebenen Funktion auszugeben.
Bei den Daten handelt es sich um den Titel, den Autor, das Erscheinungsjahr und die ISBN des Buches. Diese Daten werden dabei aus der Datei `buecherliste.txt` eingelesen.

Listing 1: buecherliste.txt

```
1 1984;George Orwell;1949;9783548267456;
2 A Scanner Darkly;Phillip K. Dick;1973;9780547572178;
3 Bodyguard;William C. Dietz;1994;9780441001057;
```

Beachte: Es müssen die folgende Datenstruktur und Funktionen implementiert werden:

- **`struct`** `_element`
- `element *insert_at_begin(element *first, element *new_elem)`
- `element *construct_element(`**`char`**` *title, `**`char`**`* author, `**`int`**` year, long long int isbn)`
- **`void`** `free_list(list *alist)`

Dagegen dürfen die `main` Funktion und die übrigen Funktionen **nicht** geändert werden. Auch dieses Mal wird wieder eine Bibliothek eingebunden (`introprog_structs_lists_input.c` und `introprog_structs_lists_input.h`), wie aus dem folgenden beispielhafter Programmaufruf ersichtlich wird:

Listing 2: Programmbeispiel

```
1  > gcc -std=c99 -Wall introprog_buecherliste.c \
2        introprog_structs_lists_input.c -o introprog_buecherliste
3  > ./introprog_buecherliste
4  Meine Bibliothek
5  ================
6
7  Buch 1
8      Titel: Mars Plus
9      Autor: Frederick Pohl
10     Jahr:  1994
11     ISBN:  9780671876050
12 Buch 2
13     Titel: Man Plus
14     Autor: Frederick Pohl
15     Jahr:  1976
16     ISBN:  9780765321787
17 Buch 3
18     Titel: Brave New World Revisited
19     Autor: Aldous Leonard Huxley
20     Jahr:  1958
21     ISBN:  9780099458234
22 [etc.]
```


Wichtig: Die Aufgabe muss den folgenden Anforderungen genügen:

- Das **`struct`** `_element` muss die folgenden Variablen beinhalten:
  - Ein **`char`** Array `title`, statisch für die Größe 255 (oder `MAX_STR`) reserviert.
  - Ein **`char`** Array `author`, statisch für die Größe 255 (oder `MAX_STR`) reserviert.
  - Eine **`int`** Variable `year`.
  - Eine **`long long int`** Variable `isbn`.
  - Ein Pointer `next` auf das nächste Element.
- Neue Elemente sollen stets an den Anfang der Liste platziert werden, sodass das neue Element jeweils die Stelle von `first` einnimmt.
- Der bestehende Code, außerhalb der geforderten Änderungen, darf nicht verändert werden. Insbesondere dürfen die Funktionen `construct_list`, `read_list` und `main` und die Datenstruktur **`struct`** `_list` (inkl. dem **`typedef`**) nicht verändert werden.
- Der Speicher soll dynamisch reserviert und restlos (auch `list *alist`) freigegeben werden.
- Im Ordner der Vorgaben liegen noch zwei weitere beispielhafte Eingabedateien, nämlich `buecherliste.evil.txt` und `buecherliste.random12342.txt`. Diese enthalten größere Beispiele. In der `buecherliste.evil.txt` ist ein Buch enthalten, welches einen sehr langen Namen (ca. 900 Zeichen) hat. Beachtet, dass wir erwarten, dass dieser Name abgeschnitten wird und bei der Ausgabe nur die ersten 254 Zeichen ausgegegeben werden. Teste dein Programm auch mit diesen Eingabedateien.
- Überprüfe mit `valgrind`, ob korrekt auf den Speicher zugegriffen wird.
- Check Deinen Code als `introprog_buecherliste.c` im SVN (im Ordner `Tutorien/t<xx>/Studierende/<tuBIT-Login>/Abgaben/Blatt04/`) ein.

Benutze die folgende Codevorgabe:

Listing 3: Vorgabe `introprog_buecherliste_vorgabe.c`

```
1  /* === INTROPROG ABGABE ===
2   * Blatt 4, Aufgabe 1
3   * Tutorium: tXX
4   * Abgabe von: Erika Mustermann
5   * ========================
6   */
7
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11 #include "introprog_structs_lists_input.h"
12
13 #define MAX_STR 255
14
15 /* Bewirkt, dass statt 'struct _element' auch 'element' verwendet werden kann
     ↪ . */
16 typedef struct _element element;
17
18 /* Bewirkt, dass statt 'struct _list' auch 'list' verwendet werden kann.
19  * Hier in einem geschrieben, so dass man auch 'list' innerhalb der
```

```c
 * struct-Definition selbst verwenden kann.
 */
typedef struct _list { /* Separater Wurzelknoten */
    element *first;    /* Anfang/Kopf der Liste */
    int count;         /* Anzahl der Elemente */
} list;

/* HIER struct _element implementieren. */

/* Fuege ein Element am Anfang der Liste an, sodass das neue Element immer
   ↪ das
 * erste Element der Liste ist.
 * Wenn die Liste leer ist soll das Element direkt an den Anfang platziert
 * werden.
 *
 * first    - Erstes Element (bzw. Anfang) der Liste
 * new_elem - Neues Element das in die Liste eingefuegt werden soll.
 *
 * Gib einen Pointer auf den neuen Anfang der Liste zurueck.
 */
element *insert_at_begin(element *first, element *new_elem) {
    /* HIER implementieren. */
}

/* Kreiere ein neues Element mit dynamischem Speicher.
 *
 * title     - Der Titel des Buches
 * author    - Autor des Buches
 * year      - Erscheinungsjahr des Buches
 * isbn      - ISBN des Buches
 *
 * Gib einen Pointer auf das neue Element zurueck.
 */
element *construct_element(char *title, char* author, int year, long long int
   ↪  isbn) {
    /* HIER implementieren. */
}

/* Gib den der Liste und all ihrer Elemente zugewiesenen Speicher frei. */
void free_list(list *alist) {
    /* HIER implementieren. */
}

/* Lese die Datei ein und fuege neue Elemente in die Liste ein
 * _Soll nicht angepasst werden_
 * */
void read_list(char* filename, list *alist) {
    element* new_elem;

    char* title;
    char* author;
    int year;
    long long int isbn;
    read_line_context ctx;
```



```c
    open_file(&ctx, filename);
    while(read_line(&ctx, &title, &author, &year, &isbn) == 0) {
        new_elem = construct_element(title, author, year, isbn);
        alist->first = insert_at_begin(alist->first, new_elem);
        alist->count++;
    }
}

/* Erstelle die Liste:
 *  - Weise ihr dynamischen Speicher zu
 *  - Initialisiere die enthaltenen Variablen
 * _Soll nicht angepasst werden_
 */
list* construct_list() {
    list *alist = malloc(sizeof(list));
    alist->first = NULL;
    alist->count = 0;
    return alist;
}

/* Gib die Liste aus:
 * _Soll nicht angepasst werden_
 */
void print_list(list *alist) {
    printf("Meine␣Bibliothek\n================\n\n");
    int counter = 1;
    element *elem = alist->first;
    while (elem != NULL) {
        printf("Buch␣%d\n", counter);
        printf("\tTitel:␣%s\n", elem->title);
        printf("\tAutor:␣%s\n", elem->author);
        printf("\tJahr:␣␣%d\n", elem->year);
        printf("\tISBN:␣␣%lld\n", elem->isbn);
        elem = elem->next;
        counter++;
    }
}

/* Main Funktion
 * _Soll nicht angepasst werden_
 */
int main(int argc, char** argv) {
    list *alist = construct_list();
    read_list(argc>1?argv[1]:"buecherliste.txt", alist);
    print_list(alist);
    free_list(alist);
    return 0;
}
```

2. Aufgabe: Eine Büchersammlung als sortierte Liste (1 Punkt) *[Einzelabgabe]*

Die Elemente sollen nun aufsteigend sortiert nach ISBN in die einfach verkettete Liste eingetragen werden. Übernimm die in der letzten Aufgabe entwickelten Funktionen. Es muss nun folgende Funktion implementiert werden:

- `element* insert_sorted(element *first, element *new_elem)`

Der Rest des Codes soll nicht angepasst werden. Auch dieses Mal wird wieder eine Bibliothek eingebunden (`introprog_structs_lists_input.c` und `introprog_structs_lists_input.h` ↪), wie aus dem folgenden beispielhafter Programmaufruf ersichtlich wird:

Listing 4: Programmbeispiel

```
> gcc -std=c99 -Wall introprog_sortierte_buecherliste.c \
                     introprog_structs_lists_input.c    \
      -o introprog_sortierte_buecherliste
> ./introprog_sortierte_buecherliste
Meine Bibliothek
================

Buch 1
    Titel: Neuromancer
    Autor: William Gibson
    Jahr:  1984
    ISBN:  9780006480419
Buch 2
    Titel: Burning Chrome
    Autor: William Gibson
    Jahr:  1986
    ISBN:  9780060539825
Buch 3
    Titel: Interface
    Autor: Neal Stephenson
    Jahr:  1994
    ISBN:  9780099427759
[etc.]
```

Wichtig: Die Aufgabe muss den folgenden Anforderungen genügen:

- Neue Elemente sollen so in die Liste eingefügt werden, dass die Elemente aufsteigend nach ISBN sortiert sind. (D.h. zu jedem Zeitpunkt gilt das erste Buch hat die kleinste ISBN und jedes darauffolgende Buch hat eine größer werdende ISBN.)
- Abgesehen von der Ordnung der Elemente gelten alle Anforderungen aus der vorherigen Aufgabe.

Wichtig: Benutze die folgende Codevorgabe, die Du auch im SVN findest.

Listing 5: Vorgabe `introprog_sortierte_buecherliste_vorgabe.c`

```
/* === INTROPROG ABGABE ===
 * Blatt 4, Aufgabe 2
 * Tutorium: tXX
 * Abgabe von: Erika Mustermann
 * ========================
 */
```



```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "introprog_structs_lists_input.h"

#define MAX_STR 255

/* Bewirkt, dass statt 'struct _element' auch 'element' verwendet werden kann
   ↪ . */
typedef struct _element element;

/* Bewirkt, dass statt 'struct _list' auch 'list' verwendet werden kann.
 * Hier in einem geschrieben, so dass man auch 'list' innerhalb der
 * struct-Definition selbst verwenden kann.
 */
typedef struct _list { /* Separater Wurzelknoten */
    element *first;    /* Anfang/Kopf der Liste */
    int count;         /* Anzahl der Elemente */
} list;

/* HIER struct _element implementieren. */

/* Fuege ein Element in die Liste ein, sodass die Liste aufsteigend nach ISBN
 * sortiert ist.
 * Dafür muss das erste Element ermittelt werden, dass in der
 * bisher sortierten Liste eine ISBN besitzt die grö ß er ist als die des
   ↪ neuen
 * Elements. Wenn die Liste leer ist soll das Element direkt an den Anfang
 * platziert werden.
 *
 * first      - Erstes Element (bzw. Anfang) der Liste
 * new_elem   - Neues Element das in die Liste eingefuegt werden soll.
 *
 * Gib einen Pointer auf den neuen oder alten Anfang der Liste zurueck.
 */
element* insert_sorted(element *first, element *new_elem) {
    /* HIER implementieren. */
}

/* Kreiere ein neues Element mit dynamischem Speicher.
 *
 * title     - Der Titel des Buches
 * author    - Autor des Buches
 * year      - Erscheinungsjahr des Buches
 * isbn      - ISBN des Buches
 *
 * Gib einen Pointer auf das neue Element zurueck.
 */
element *construct_element(char *title, char* author, int year, long long int
   ↪  isbn) {
    /* HIER implementieren. */
}

```

```c
/* Gib den der Liste und all ihrer Elemente zugewiesenen Speicher frei. */
void free_list(list *alist) {
    /* HIER implementieren. */
}

/* Lese die Datei ein und fuege neue Elemente in die Liste ein
 * _Soll nicht angepasst werden_
 * */
void read_list(char* filename, list *alist) {
    element* new_elem;
    char* title;
    char* author;
    int year;
    long long int isbn;
    read_line_context ctx;
    open_file(&ctx, filename);
    while(read_line(&ctx, &title, &author, &year, &isbn) == 0) {
        new_elem = construct_element(title, author, year, isbn);
        alist->first = insert_sorted(alist->first, new_elem);
        alist->count++;
    }
}

/* Erstelle die Liste:
 *  - Weise ihr dynamischen Speicher zu
 *  - Initialisiere die enthaltenen Variablen
 * _Soll nicht angepasst werden_
 */
list* construct_list() {
    list *alist = malloc(sizeof(list));
    alist->first = NULL;
    alist->count = 0;
    return alist;
}

/* Gib die Liste aus:
 * _Soll nicht angepasst werden_
 */
void print_list(list *alist) {
    printf("Meine Bibliothek\n================\n\n");
    int counter = 1;
    element *elem = alist->first;
    while (elem != NULL) {
        printf("Buch %d\n", counter);
        printf("\tTitel: %s\n", elem->title);
        printf("\tAutor: %s\n", elem->author);
        printf("\tJahr:  %d\n", elem->year);
        printf("\tISBN:  %lld\n", elem->isbn);
        elem = elem->next;
        counter++;
    }
}

/* Main Funktion
 * _Soll nicht angepasst werden_
 */
int main(int argc, char** argv) {
    list *alist = construct_list();
    read_list(argc>1?argv[1]:"buecherliste.txt", alist);
    print_list(alist);
    free_list(alist);
    return 0;
}
```