Fakultät IV Elektrotechnik und Informatik
Einführung in die Programmierung WS 2019/2020
Manfred Hauswirth / Damien Foucard / Christoph Seifert

Aufgabenblatt 4

Ausgabe: *29.11.2019*
Abgabe: *10.12.2019 09:59*

Thema: *Listen, Datenstrukturen*

Abgabemodalitäten

1. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf den Rechnern des ZE-CM/eecsIT mittels `clang -std=c11 -Wall -g` kompilieren.
2. Die Abgabe erfolgt ausschließlich über unser SVN im Abgaben-Ordner. Nur wenn ein Test in Osiris angezeigt wird ist sichergestellt, dass die Abgabe erfolgt ist.
3. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben.
4. Die Abgabe erfolgt in folgendem Unterordner:
   `introprog-wise1920/Studierende/<L>/<TU-Login>/Abgaben/Blatt<XX>`
   wobei `<L>` durch den ersten Buchstabe des TU-Logins und `<XX>` durch die Nummer des Aufgabenblattes zu ersetzen sind. Die Ordner werden automatisch angelegt sobald die Abgabe freigeschaltet wird.
5. Du darfst den Abgabeordner für das Blatt nicht selbst erstellen. Die Ordner erstellen wir kurz nach der Ausgabe der Aufgabe.
6. Um die Änderungen (z. B. neue Abgabeordner) vom Server abzuholen, musst Du den Befehl `svn update` im Wurzelverzeichnis (also im Verzeichnis `introprog-wise1920`) des Repositories ausführen.
7. Im Abgabeordner gelten einige restriktive Regeln. Dort ist nur das Einchecken von Dateien mit den in der Aufgabe vorgegebenen Namen erlaubt, ausserdem werden die Abgabefristen vom Server überwacht. Beachte eventuelle Fehlermeldungen beim SVN-Commit. Lade nur Dateien hoch, die Du selbst bearbeiten sollst, insbesondere also keine Vorgaben.
8. Der Dateiname bei Abgaben mit C-Code soll dem der Vorgaben entsprechen. Entferne ausschließlich `_vorgabe` aus dem Name. Bei Textaufgaben sind, wenn nicht anders angegeben, `.txt` Dateien zugelassen, die als Plaintext-Datei zu speichern sind (keine Word-Dateien umbenennen).
9. Es gibt einen Ordner `Arbeitsverzeichnis`, in dem Du Dateien für Dich ablegen kannst.
10. Die Ergebnisse der automatischen Tests kannst Du auf OSIRIS einsehen:
    https://osiris.ods.tu-berlin.de/

Aufgabe 1 Eine Büchersammlung als Liste (7 Punkte)

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

```
1984;George Orwell;1949;9783548267456;
A Scanner Darkly;Phillip K. Dick;1973;9780547572178;
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 hier wird wieder eine Bibliothek eingebunden (`introprog_structs_lists_input.c` und `introprog_structs_lists_input.h`), wie aus dem folgenden beispielhaften Programmaufruf ersichtlich wird:

Listing 2: Programmbeispiel

```
> clang -std=c11 -Wall introprog_buecherliste.c \
      introprog_structs_lists_input.c -o introprog_buecherliste
> ./introprog_buecherliste
Meine Bibliothek
================

Buch 1
    Titel: Mars Plus
    Autor: Frederick Pohl
    Jahr:  1994
    ISBN:  9780671876050
Buch 2
    Titel: Man Plus
    Autor: Frederick Pohl
    Jahr:  1976
    ISBN:  9780765321787
```

```
Buch 3
    Titel: Brave New World Revisited
    Autor: Aldous Leonard Huxley
    Jahr:  1958
    ISBN:  9780099458234
[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 `introprog-wise1920/Studierende/<L>/<TU-Login>/Abgaben/Blatt04`) ein.

Benutze die folgende Codevorgabe:

Listing 3: Vorgabe `introprog_buecherliste_vorgabe.c`

```c
#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 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. */
}
```

```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_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;
}
```

Aufgabe 2 Eine Büchersammlung als sortierte Liste (3 Punkte)

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 zusätzlich implementiert werden:

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

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

Listing 4: Programmbeispiel

```
> clang -std=c11 -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`

```c
#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 größer 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. */
}

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

```c
}

/* 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;
}
```