Verwendung der Liste

Zum Benutzen der beigelegten Liste muss `lists.h` inkludiert werden. Im folgenden Beispiel werden Instanzen der struct `Student` erstellt und in eine Liste eingefügt. Danach wird der zweite Student gelöscht, die Anzahl verbliebener Studenten ausgegeben, die Liste nach dem Alter der Studenten aufsteigend sortiert und ausgegeben. Zum Schluss wird die Liste gelöscht.

Die Elemente, die in der Liste gehalten werden, sind structs, die das Makro `LIST_NODE_HEADER` beinhalten (dadurch bekommen sie automatisch einen `next`- und `previous`-Pointer auf gleichartige structs und werden zum Listenknoten) mit ihren Daten. Die Liste selbst ist ein struct, der das Makro `LIST_HEADER` beinhaltet.

Die Liste kann sortiert werden, indem der Funktion `List_sort` ein Funktions-Pointer auf eine Sortierfunktion der Signatur `int sortFunction(const void *node1, const void *node2, void *userData)` übergeben wird. In dieser Funktion werden zwei Knoten anhand eines bestimmten Merkmals (hier Alter) verglichen. Ist der erste Knoten in Bezug auf das Merkmal größer als der zweite, wird 1 zurückgegeben. Ist er kleiner, wird -1 zurückgegeben und bei Gleichheit 0.

```
#include "lists.h"

typedef struct Student
{
    LIST_NODE_HEADER(struct Student);
    char *name;
    int alter;
} StudentNode;

typedef struct
{
    LIST_HEADER(StudentNode);
} StudentList;

int compareStudentNodes(StudentNode *node1, StudentNode *node2, void *userData)
{
    if (node1->alter > node2->alter)
        return 1;
    else if (node1->alter < node2->alter)
        return -1;
    else
        return 0;
}
```

Prof. Dr.-Ing. Martin Hoffmann
martin.hoffmann@fh-bielefeld.de

```
static StudentList list;

int main(void)
{
    // Liste initialisieren
    List_init(&list);

    // Studenten erstellen
    StudentNode *a = LIST_NEW_NODE(StudentNode);
    a->name = "Wilfried";
    a->alter = 22;

    StudentNode *b = LIST_NEW_NODE(StudentNode);
    b->name = "Schnabelbert";
    b->alter = 92;

    StudentNode *c = LIST_NEW_NODE(StudentNode);
    c->name = "Hubert";
    c->alter = 12;

    // Studenten zur Liste hinzufügen
    List_append(&list, a);
    List_append(&list, b);
    List_append(&list, c);
    // b löschen und Anzahl der Studenten ausgeben
    List_remove(&list, b);
    printf("Anzahl der Studenten: %d", List_count(&list));

    // Liste nach Alter sortieren
    List_sort(&list, (ListNodeCompareFunction) compareStudentNodes, NULL);

    // Jedes Element der Liste ausgeben
    StudentNode *current = list.head;
    while (current != NULL)
    {
        printf("%s - %d\n", current->name, current->alter);
            current = current->next;
    }

    // Alle Nodes löschen (reicht nur für Nodes, die keine Pointer auf allozierten
    // Speicher besitzen)
    List_done(&list, NULL, NULL);
    return 0;
}
```