# Prozedurale Programmierung – Übung 12

*WS 2022/23* *Prof. Dr. Johannes Jurgovsky*

Technische Hochschule Rosenheim

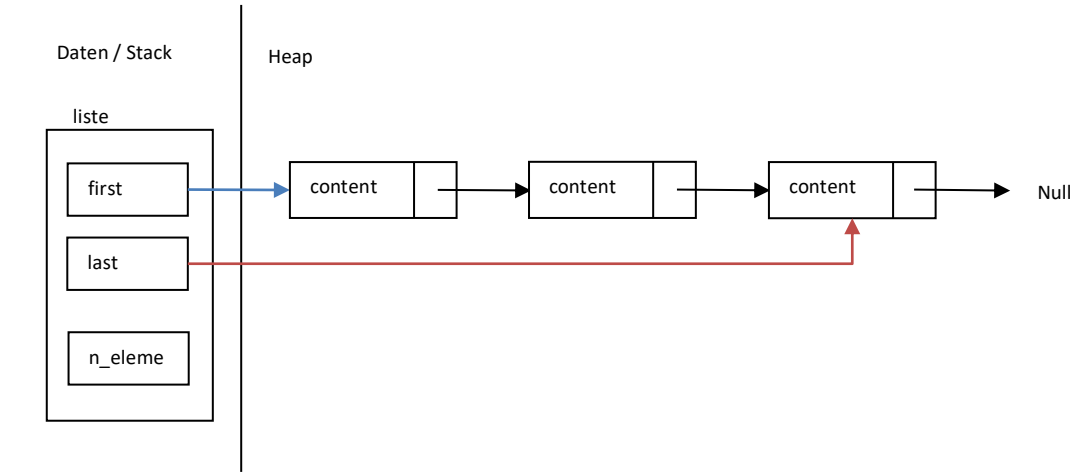
In der Community wird ein Projekt in der Datei „listen.zip“ bereitgestellt.

Als Beispiel für eine allgemeine (einfach verkettete) Listenverwaltung bauen wir einen ADT (abstract data type) „Liste“, der auf folgender Datenstruktur beruht (siehe Datei list.h)

```
#define LENGTH 80

typedef struct s_Element
{
   char content[LENGTH];       // Inhalt als String fester Länge
   struct s_Element *next;     // Zeiger auf den Nachfolger
} t_Element;

typedef struct
{
   struct s_Element *firstElement;
   struct s_Element *lastElement;
   int n_elements;
} t_List;
```




## Aufgabe 1

Das Projekt enthält u.a. die Header-Datei ***list.h*** und die Source-Datei ***list.c*** mit den Funktionen `initList(...)` und `pushFront(...)`. Erweitern Sie die Datei ***list.c*** so, dass diese zusätzlich folgende Funktionen bereitstellt (die Prototypen sind bereits im Header bzw. der Source-Datei enthalten):

```
// gibt die Liste ‚list‘ am Bildschirm aus
void printList(const t_List *list);

// gibt den Inhalt des ersten Elements zurück (kopiert den string in den
// übergebenen Buffer s) und entfernt das Element aus ‚list‘
void popFront(t_List *list, char *s);

// erzeugt ein neues Element und haengt es an das Ende der Liste ‚list‘ an
void pushBack(t_List *list, const char *s);
```

Testen Sie die neuen Funktionen, indem Sie die Funktion `run_tests()` der Testumgebung in ***tester.c*** aufrufen. Die Testumgebung prüft die korrekte Funktion Ihrer Implementierung.

## Aufgabe 2

Implementieren Sie eine Funktion `bool saveContent(List_t * list, char * fileName)`, die die Inhalte der Liste in eine Datei mit dem Namen `fileName` schreibt (und ersetzt, falls die Datei bereits existiert).
Die Zeichenkette jedes Elements soll in einer neuen Zeile gespeichert werden.