Instructions - Laboratory assignment 4 WS 22

Johan Holmgren, 2018-09-19 - Created assignment
Johan Holmgren, 2020-09-14 - Modification to work with VSCode, PlatformIO, and ESP32 board.

Overview and purpose of task

The purpose of this assignment is to learn how to implement a linked list data structure.

A linked list can be seen as a series of elements (or objects) that are linked to each other, thus forming a chain of elements.
Each element contains a data field (which could be a complex structure) and one or more pointers to other elements. In this assignment you will implement two types of linked lists.

In addition, you will formulate and implement your own test cases using the knowledge you gained in the previous course assignments.

You will conduct this assignment either individually or in pairs of two students; however please note that the examination is individual.

Introduction to linked lists

A linked list is a data structure that contains a chain of elements that are linked to each other in sequence.

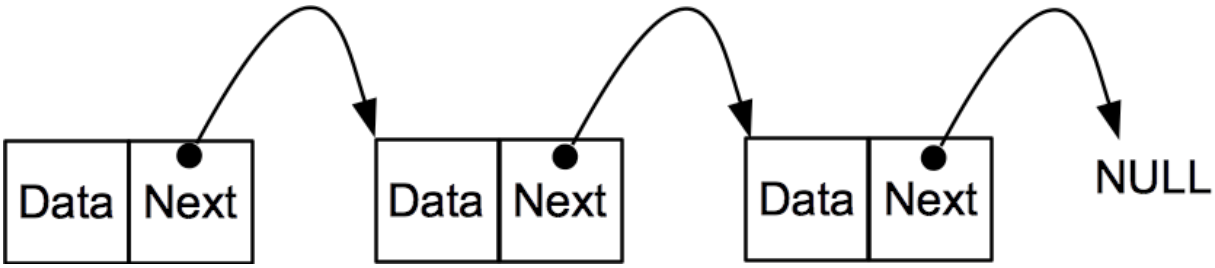
There are different types of linked lists; however, a common characteristic of linked lists is that you access elements by traversing the chain of linked elements until you find the element you are looking for. For example, if you want to access an element somewhere in the chain of elements, you first need to visit the first element in the chain to get access to the second element. Then you visit the second element to get access to the third element. Then you continue in this way until you find the element you want to access. When you want to insert an element somewhere in the linked list, you need to search through the linked list in the same way.

In this assignment you will work with two types of linked lists: *single linked lists* and *double linked lists*, each of which are described below.

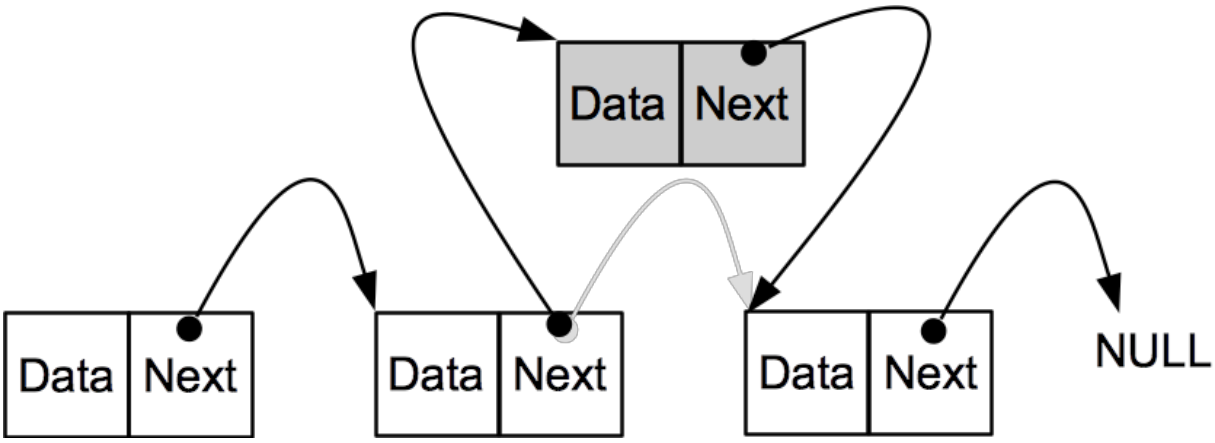
Single linked list

A single linked list is a chain of elements, where each of the elements, except for the last element, in the list has a pointer `next` to the next element in the list. The last element points to nothing (or `NULL`) in order to indicate that it is the end of the list.

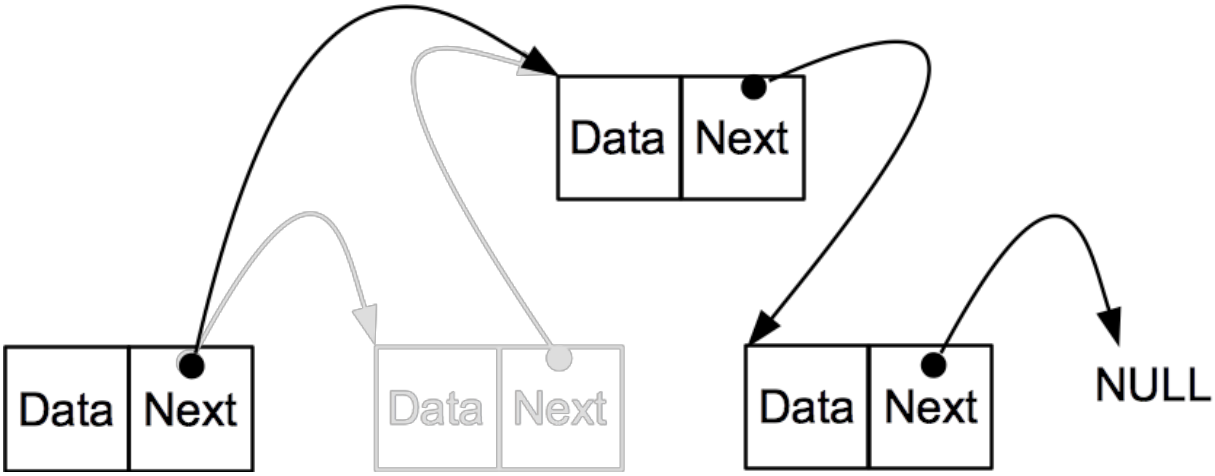
The following picture illustrates a single linked list containing three elements.



The following image shows what will happen when a new element (in grey) is added between the second element and the third element in a single linked list. Please note what happens with the `next` pointers of the second and the new element.



The following image shows what will happen when the second element is removed from the list. Please note what happens with the `next` pointer of the first element.

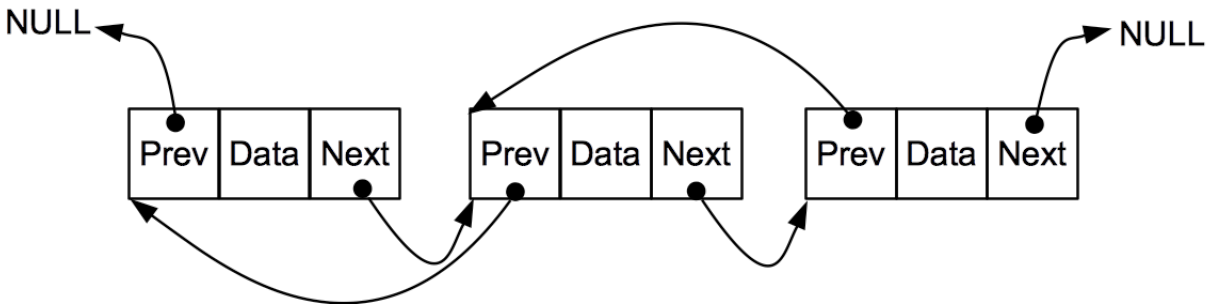


Double linked list

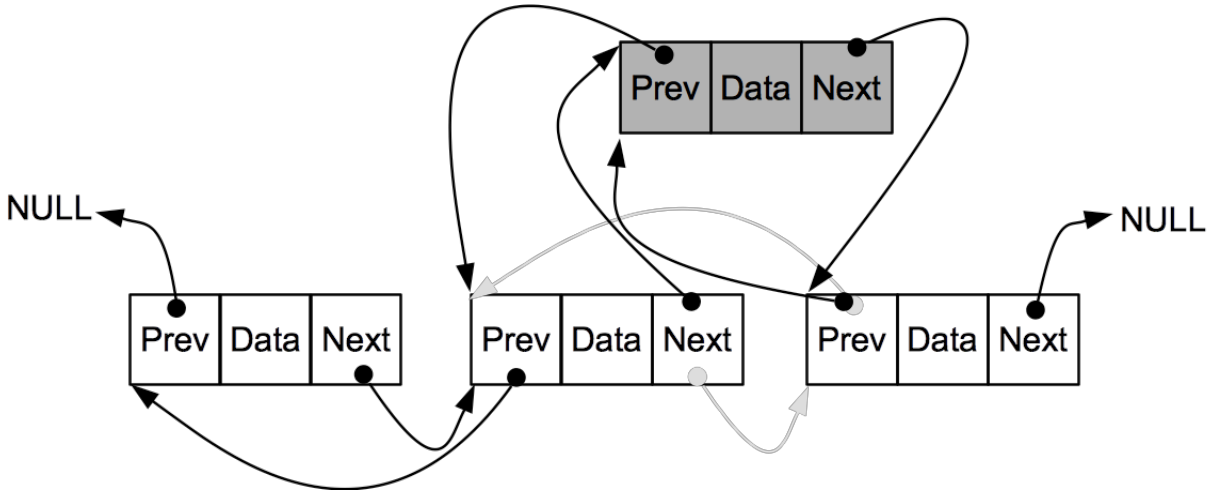
A double linked list is a chain of elements, where each of the elements, except for the first and last element, in the list has a pointer (`previous`) to the previous element in the list and a pointer (`next`) to the next element in the list.

For the last element in the list, `next` points to nothing (`NULL`) and for the first element in the list, (`previous`) points to nothing (`NULL`).

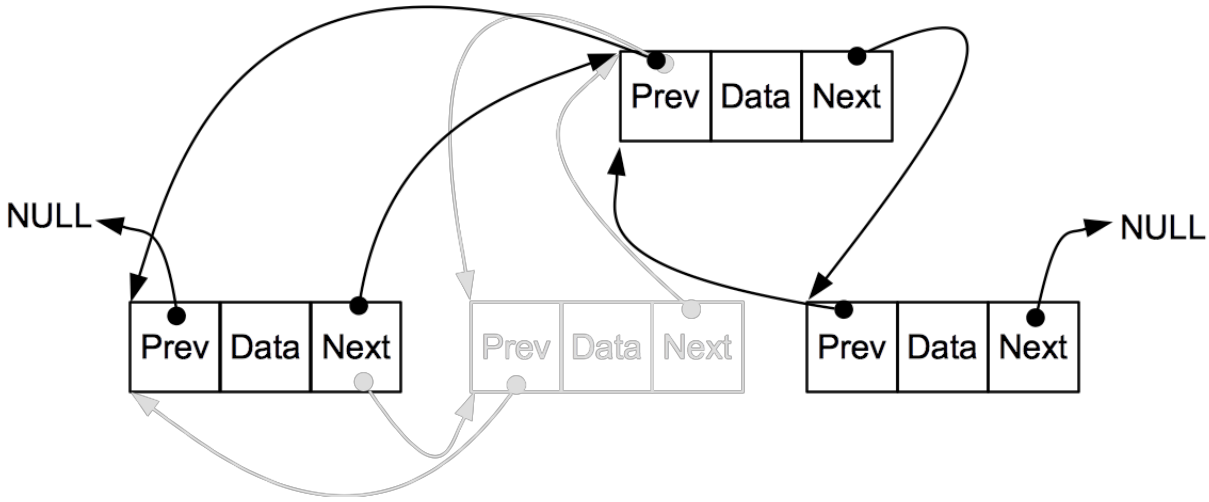
The following image illustrates a double linked list containing three elements.



The following image illustrates what happens when you insert an element (between the second and third elements) in a double linked list containing three elements. Please study what happens with the pointers of the new element, and the second and third elements.



The following image illustrates what happens when you remove an element (the second element) in a double linked list containing four elements. Please study what happens with the pointers of the new element, and the second and third elements.



Description of task

Your task in this laboratory assignment is to implement several functions that allow you to work with a single linked list and with a double linked list. The elements should contain an integer (data) value that you will use to sort the elements. This means that when you add a new element in the linked list, you should make sure that you maintain a sorted linked list. The first element is the smallest element in the list, the second element is the second smallest element in the list, and so on. The last element is the largest element in the list.

To help you get started, you are provided a visual studio code project using PlatformIO with the following settings:

- platform: espressif32
- board: nodemcu-32s
- framework: espidf

- monitor_speed: 115200

You find the "startup code" in a zip file on Canvas. To get started, you import the existing project into visual studio code. The project contains the following source files:

- `single_linked_list.h` - Header file specifying all the functions you should implement for a single linked list.
- `single_linked_list.c` - The file where you implement the functions specified in `single_linked_list.h`
- `double_linked_list.h` - Header file specifying all the functions you should implement for a double linked list.
- `double_linked_list.c` - The file where you implement the functions specified in `double_linked_list.h`
- `main.c` - The file where you write the code to test your linked lists.

Please note that the code should run on your ESP32 board, and you should illustrate that your linked lists work correctly by applying all the implemented functions, and carefully test your functions according to the tests you create.

Implement a single and a double linked list

As mentioned above, you are provided two c header files `single_linked_list.h` and `double_linked_list.h`, which contains a single linked list data structure and a double linked list data structure, respectively.

The single linked list data structure is defined in the following way:

```
struct singleLinkedList{
  struct singleLinkedListElement *first;
};
```

The elements in a single linked list are defined in the following way:

```
struct singleLinkedListElement{
  int data;
  struct singleLinkedListElement *next;
};
```

The double linked list data structure is defined in the following way:

```
struct doubleLinkedList{
  struct doubleLinkedListElement *first;
  struct doubleLinkedListElement *last;
};
```

The elements in a double linked list are defined in the following way:

```
struct doubleLinkedListElement{
  int data;
  struct doubleLinkedListElement *next;
  struct doubleLinkedListElement *previous;
};
```

Each of the `single_linked_list.h` and `double_linked_list.h` header files contains a set of functions that you will implement in this assignment.

`single_linked_list.h` contains the following functions:

- `int addElementSingleLinkedList()`
- `void initSingleLinkedList()`
- `int removeFirstElementSingleLinkedList()`
- `int removeLastElementSingleLinkedList()`

`double_linked_list.h` contains the following functions:

- `int addElementDoubleLinkedList()`
- `void initDoubleLinkedList()`
- `int removeFirstElementDoubleLinkedList()`
- `int removeLastElementDoubleLinkedList()`

Specifications of each of these functions are provided in the provided header files.

In this assignment you are expected to use the standard c memory management functions `malloc()`, `calloc()`, and `free()`, defined in `stdlib.h`, when allocating and releasing memory. In an embedded system, which is typically very memory constrained and you need to be careful about which part of the memory you use for different purposes, this is typically not how you would manage memory. However, in this assignment we want you to practice memory management using the standard c approach (using `malloc()`, `calloc()`, and `free()`).

Please note that you are recommended to test, using pen and paper, what will happen with your linked lists when you run the operations you will implement in your functions. This might help you understand how to write your code.

Implement test cases

As part of this assignment, you should implement a set of test cases (both black box and white box), similar to what you have done previously. The difference now is that you should both identify and implement a set of relevant test cases for your linked lists.

Submission and examination

When you are done with the assignment, you should upload the `src` folder of your visual studio code project (in a zip file) using the submission page for assignment 4 on Canvas.

You should also demonstrate and discuss your solution with the examiner of assignment 4, which you do during one of the scheduled presentation times for laboratory assignment 4 (examination times will be published on Canvas). During the discussion, you should be able to show that your code works correctly, explain the code, and answer some basic questions about your solution.