Lab Course (Basics & Lab)

Universität Bonn
Institut für Informatik II
June 6, 2016

Summer term 2016
Prof. Dr. Reinhard Klein
Stefan Krumpen, Alexander Dieckmann

Sheet 6 - Linked Lists

Practical assignments are corrected in the exercises on *Mo 13/6 14:30*
Single submissions!

Assignment 1 (Linked List, *4 points*)

In that assignment you will get familiar with basic list data structures. Similar to arrays linked lists[1] represent a sequence of data elements. But in contrast to arrays the elements are not stored sequential in memory, but are instead connected by pointers to the previous and next element (double linked list). For the first element `_previous` and for the last element `_next` is `NULL`.

```
struct Element {
    Element* _previous;
    Element* _next;
    std::string value;
};

class LinkedList {
private:
    Element* _head;
    Element* _tail;
    unsigned int number_of_elements;

public:
    LinkedList();
    ~LinkedList();

    Element* head();     // return first element
    Element* tail();     // return last element
    unsigned int size(); // return number of elements

    Element* remove(Element* element); // remove element e, return the previous element
    Element* insert(Element* previous, std::string value);
    // add value after 'previous'; if 'previous' is 'NULL' prepend; return the new element

    void clear();  // clear elements
    void print();  // print contents
};
```

a) Write an implementation for the interface.

b) We will now test your list and expand it with a few useful functions. In each step of the following exercises, print the list contents after *each* step!

c) Create a list and add the following animals at the end: `wolverine`, `chicken`, `turtle`, `dog`, `snail`, `cat`, `sardine`, `zebra`, `dog`, `snail`, `dog`. Finally clear the list (of course this must release all allocated memory).

d) Write a function `unsigned int remove_value(std::string value)`, that removes all strings that equal to value and returns their count. It should only use public class methods. Add the previous animals to an empty list, then remove `dog` and `wolverine`.

[1] `http://en.wikipedia.org/wiki/Linked_list`

Note: In order to compare two different strings, you can use the function `str1.compare(str2)`. The documentation can be found at here[2]

e) Write a function `void insert_sorted(std::string value)` that inserts a string before the first element that does not compare smaller (use lexicographic comparison). It should only use public class methods. Again add the animals to an empty list. This time in each step the list should be sorted.

f) Add the animals to a sorted list. Now query the console for user input. If the input is already contained, remove all elements equal to the input. Otherwise add it at the right position. Output the list at each step.

g) Give the algorithmic complexity for

- inserting front/back
- removing front/back
- inserting sorted
- remove by value
- accessing by index

h) What do you conclude are advantages/disadvantages of arrays and lists ?

Good luck !

[2] `http://www.cplusplus.com/reference/string/string/compare/`