Assignment 4

Forfall 18. sep. 2020 av 23.59 **Poeng** 100 **Tilgjengelig** til 4. okt. 2020 i 23.59

Denne oppgaven ble låst 4. okt. 2020 i 23.59.

Introduction

The purpose of this hand-in is to give a practical introduction to pointers and memory management together with templates. The code won’t be very large, the challenge is tied to understanding how the allocated memory fits together.

4.1

Design and implement a templated linked list, meaning a class that can store any number of objects of any given type. The internal implementation is up to you, but if you want a pointer in the right direction I am partial to double linked lists.

If you do not know where to start, read this:
https://en.wikipedia.org/wiki/Linked_list (https://en.wikipedia.org/wiki/Linked_list)

Create a class for the list and a class for the node. You must then figure out how to add a new node when the user calls push_back(), and how to remove a node when the user calls remove().

You are not allowed to reuse any of the existing list implementations such as those in STL or to use arrays in the template class.

The list must support the following:

- void push_back(const T& value) Add a value to the back of your list
- void remove(std::size_t position) Remove item at specified position
- std::size_t size() Get the item count
- T& operator[](std::size_t position) Retrieve item by index

Areas of interest: Pointers, new/delete operators, template classes

Optional: If you want you can create your own iterator class to support container iteration. This would complement or replace operator[]. This is not mandatory.

Application UI

Write an application that has a list of integers and a list of strings (using your template list class, so you only write one class that can support **any** type), and has this menu:

1. Add integer (asks for one, then displays the menu again)
2. Show integer at position (ask for position)
3. Remove integer at position (ask for position)
4. Add string (asks for one, then displays the menu again)
5. Show string at position (ask for position)
6. Remove string (ask for position)
7. Print information (see below)
8. Exit

Note: Positions start from 0.

Expected output (from 2 and 5):
Integer in position 5: 42
String in position 3: Hello, World!

Expected output (from 7):
Got 4 integers: 1 2 3 4
Got 4 strings: one two three four

4.2

Document the complexity of each function in a comment above the declaration:
http://en.wikipedia.org/wiki/Time_complexity **(http://en.wikipedia.org/wiki/Time_complexity)**

Delivery

1. Test locally

see *Local testing* in Canvas

Test the solution on your own computer. You can fix your code and retest as many times as you want.

2. Deliver and test in Bamboo

See *Delivery and Bamboo testing* in Canvas

Finally deliver and test the solution in Bamboo. This is what updates your score in Canvas which is the score I can see. You can test as many times as you want here as well.