Skilaverkefni III

Lýsing á íslensku

Í þessu verkefni eigið þið að útfæra hugræna gagnatagið `LinkedList` (tvítengdan lista) sem sniðmátsklasa. Grunnurinn að klasanum (`Node.h` og `LinkedList.h`) er í möppunni `Verkefni` í repo https://github.com/reykjavik-university/2016-T-201-GSKI/. Aðalforritið `main.cpp`, sem prófar útfærsluna, er einnig gefið. Þið eigið því eingöngu að útfæra `LinkedList.cpp` og megið ekki breyta hinum skránum.

Til að einfalda útfærsluna (þ.e. fækka sérstökum tilfellum) eigið þið að bæta sérstökum hnútum (svokölluðum „sentinel nodes“) við báða enda tvítengda listans: `header` hnút á undan fyrsta staki listans og `trailer` hnút á eftir síðasta staki listans. Lesið um „sentinel nodes“ í kafla 3.3 í kennslubókinni eftir Goodrich og/eða kafla 4.1.5 í kennslubókinni eftir Shaffer.

Í útfærslunni eigið þið að gera greinarmun á síðasta hnúti listans og enda listans. Endir listans er `trailer` en síðasta hnútur listans er hnúturinn næst á undan `trailer`. Byrjun listans er aftur á móti fyrsti hnútur, þ.e. hnúturinn á eftir `header`. `LinkedList` er með meðlimabreytuna `currNode` sem er bendir á núverandi hnút. `currNode` getur verið á bilinu <fyrsti hnútur, `trailer`>, þ.e. `currNode` bendir aldrei á `header`.

Aðalforritið les fyrst inn heiltölu, `count`, segir til um hversu mörg stök fylgja á eftir og síðan `count` gildi sem mynda stök listans. Ofangreint er fyrst gert fyrir heiltölulista og síðan fyrir strengjalista.

Í tengslum við þetta verkefni er gott fyrir ykkur að hlusta að fyrirlesturinn um „Minnisvillur“ sem aðgengilegur er undir „Annað efni“ í Myschool.

English description

In this project, you need to implement the abstract data type `LinkdedList` (a doubly linked list) as a template class. The base for the class (`Node.h` og `LinkedList.h`) is in the folder `Verkefni` in the repo https://github.com/reykjavik-university/2016-T-201-GSKI/. The main program, `main.cpp`, which tests the implementation, is also given. Therefore, you only need to implement `LinkedList.cpp` and are not allowed to change the other files.

To simplify the implementation (i.e. reduce special cases) you need to add so-called sentinel nodes at both ends of the doulby linked list: a `header` node before the first node of the list and a `trailer` node after the last element of the list. Read about sentinel nodes in chapter 3.3 in the textbook by Goodrich and/or chapter 4.1.5 in the textbook by Shaffer.

In the implementation you should distinguish between the last node of the list and the end of the list. The end of the list is the `trailer`, whereas the last node is the element preceeding the `trailer`. On the other hand, the start of the list is the first node, i.e. the node following the `header`. `LinkedList` has the member variable `currNode` which points to the current node. `currNode` can range from <first node, `trailer`>, i.e. it never points to the `header`.

The main program first reads an integer, `count`, denoting how many elements follow, and then `count` values are read which comprise the list elements. The above is first carried out for a list of integers and then a list of strings.

Um meðlimaföll í `LinkedList` / About member functions in `LinkedList`

- `next()`: moves `currNode` to the next node given that `currNode` is not already the `trailer`.
- `prev()`: moves `currNode` to the previous node given that `currNode` is not already the first node
- `moveToEnd()`: moves `currNode` past the last node of the list, i.e. to the `trailer`
- `moveToPos()`: moves `currNode` to position pos (the first node of the list has position 0). If moving to a position which is not in the list, then moves `currNode` to the trailer.
- `insert()`: inserts an element before `currNode` without affecting to which `currNode` points
- `remove()`: removes `currNode` and returns its element. Makes `currNode` point to the node following the one being removed.
- `clear()`: removes all nodes, except `header` and `trailer`, and initializes all member variables
- `removeAll()`: removes all nodes, except `header` and `trailer`
- `init()`: initializes all member variables
- `LinkedList(LinkedList<T>& lis)`: Copy constructor; initializes the list with copies of data from lis.

Dæmi um inntak og úttak / Example input and output:

Printing empty list:

Now reading list data:
5
1 2 3 4 5

List length is: 5
1
2
3
4
5

Removed element: 3
List length is: 4
Current element: 4
1
2
4
5

Last element: 5
List length is: 6
1
2
1
5
4
5

Last element: 5
1
1
1
2
1
5
4

```
5
5

The list is empty
List length is: 2
1
5

Copying list:
1
5

1
5
```