T-201-GSKI, GAGNASKIPAN

Vor 2017

INHERITANCE - QUEUE/STACK

Assignment grading. For full marks your solution needs to be accepted by Mooshak. Points may be deducted for

- not implementing the solution according to the specifications,
- redundant or repeated code.
- messy code,
- code that is difficult to understand and not explained in comments.

If your solution is not accepted by Mooshak, it will receive a maximum grade of 7.

QUEUE/STACK and LINKEDLIST

Your assignment is to implement a **Queue** class and a **Stack** class using a **Single-Linked List** data structure. The classes Queue and Stack should be abstract classes with no implementation, while the class LinkedList should inherit them both and implement the functionality to actually store the data entered into them.

You can begin your implementation using either your own, or the teacher's solution to the previous LinkedList project.

THE IMPLEMENTATION

All classes, Queue, Stack and LinkedList are implemented as template classes, so they can be tested for different data types. Remember that the simplest way to get template classes to compile correctly is to implement them entirely within the .h header files.

Queue and Stack are both abstract classes, with only pure virtual functions.

There is one exception to this, which is the stream << overload which is a friend function and can't be virtual. In order to make that functionality virtual we implement a pure virtual print() function which the ostream from the << overload is sent into. The print function can now be implemented in derived classes like any other virtual function.

The class LinkedList must now inherit both classes, Stack and Queue (make sure you forward the template <T> correctly) and implement their pure virtual functions. Queue has the functions add() and remove() and Stack has the functions push() and pop() and they both have the function print(). All of these must be implemented in LinkedList.
The functionality behind them can be as simple as calling functions you already have on the linked list, push_front, push_back and pop_front (or HeadAdd, TailAdd and HeadRemove).

RÖÐ/STAFLI og TENGDUR LISTI

Verkefnið er að útfæra **röð** og **stafla** sem nota **eintengdum lista** til að útfæra virknina. Klasarnir Queue og Stack eiga að vera abstract klasar með engar útfærslur, á meðan klasinn LinkedList á að erfa þá báða og útfæra virknina til að geyma gögnin sem bætt er í þá.
Þið egið byggja LinkedList útfærsluna ykkar hvort sem er á ykkar eigin eða lausn kennara við fyrra LinkedList verkefni.

ÚTFÆRSLA

Allir klasarnir, Queue, Stack og LinkedList eru template klasar, þannig að þeir geta verið prófaðir með mörgum gagnatýpum. Athugið að einfaldast er að útfæra **template** klasa í heild sinni inni í .h skrám.
Queue og Stack eru báðir abstract klasar, sem þýðir að þeir eru einungis með “pure virtual” föllum. Eina undantekningin er yfirskriftin á straumoperatornum <<, sem er friend fall og getur því ekki verið virtual. Til þess að sú virkni sé virtual kallar operator fallið á fall sem heitir print(), sendir strauminn áfram í það. print() fallið er skilgreint pure virtual og því getur sú virkni verið útfærð inni í undirklösum eins og hvert annað virtual fall.
Klasinn LinkedList verður að erfa báða klasana, Stack og Queue (athugið að senda template <T> rétt áfram) og útfæra virtual föllin þeirra. Queue hefur föllin add() og remove() og Stack hefur föllin push() og pop() og þeir hafa báðir fallið print(). Öll þessi föll verða að vera útfærð í LinkedList.

Virknin á bak við þau getur einfaldlega verið að kalla á föll sem eru þegar til á tengda listanum, push_front, push_back og pop_front (eða HeadAdd, TailAdd og HeadRemove).