Mètode de la classe llista per a revessar-la X97433_ca

Implementeu un nou mètode de la classe List que revessi els seus propis elements. És a dir, el que estava al principi ara apareixerà al final, el que era el segon des del principi ara apareixerà com a segon des del final, i així successivament.
D'entre els fitxers que s'adjunten en aquest exercici, trobareu `list.hh`, a on hi ha una implementació de la classe genèrica `List`. Haureu de buscar dins `list.hh` la part:

```
// Pre: Sigui [e1,e2...,en] el contingut inicial de la llista des del principi
// Post: El contingut final de la llista és [en,...,e2,e1], és a dir, la llista
// Descomenteu les següents dues linies i implementeu la funció:
// void reverse() {
// }
```

Haureu de descomentar les dues línies que s'indiquen i implementar aquest mètode. No toqueu la resta de la implementació de la classe, excepte si, per algun motiu, considereu que necessiteu afegir algun mètode auxiliar a la part privada.
Preferiblement, haurieu d'aconseguir implementar `reverse` a base d'intercanviar els punters de l'objecte. Una implementació alternativa provocarà error d'execució perquè els jocs de proves estaran treballant amb iteradors sobre els elements de la llista.
D'entre els fitxers que s'adjunten a l'exercici també hi ha `main.cc` (programa principal), i el podeu compilar directament, doncs inclou `list.hh`. Només cal que pugeu `list.hh` al jutge.

Entrada

La entrada del programa és una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre una llista que se suposa inicialment buida i un iterador que se suposa situat inicialment al principi (i final) d'aquesta llista:

```
push_front s (s és un string)
push_back s (s és un string)
insert s (s és un string)
pop_front
pop_back
it++
it--
*it
reverse
```

Se suposa que la seqüència d'entrada serà correcta (sense `pop_front` ni `pop_back` sobre llista buida, ni `*it` tenint `it` situat al `end` de la llista). Tampoc hi haurà `pop_front` just quan l'iterador estigui apuntant al primer element de la llista, ni hi haurà `pop_back` just quan l'iterador estigui apuntant a l'últim element de la llista (tingueu en compte que l'últim element de la llista no és el `end` de la llista).
El programa principal que us oferim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe list. Només cal que implementeu els mètodes abans esmentats.

Sortida

Per a cada instrucció `*it`, s'escriurà el contingut apuntat per l'iterador. El programa que us oferim ja fa això. Només cal que implementeu el mètode abans esmentat.

Exemple d'entrada 1

```
reverse
insert a
reverse
it--
*it
push_back b
reverse
it--
*it
it++
reverse
*it
push_front c
reverse
it++
*it
push_back d
reverse
it--
*it
push_back e
reverse
*it
it--
*it
it--
reverse
it--
*it
```

Exemple de sortida 1

```
a
b
a
c
d
d
c
c
```

Exemple d'entrada 2

```
insert de
reverse
it--
push_back q
reverse
reverse
push_front e
push_front p
push_back tf
it--
pop_front
*it
push_back rq
push_front s
it--
it++
it--
push_back s
push_back l
*it
it++
pop_front
push_back u
it++
pop_back
push_back gs
push_front ok
push_front s
it++
reverse
it++
*it
insert j
pop_front
push_back xg
push_back lo
push_back j
pop_back
reverse
push_back i
push_front ir
reverse
*it
insert wg
pop_back
```

```
pop_front
*it
push_front fz
reverse
push_front hh
push_back lr
*it
*it
it++
pop_front
reverse
*it
pop_front
pop_back
push_front y
*it
```

Exemple de sortida 2

```
e
s
de
de
de
de
de
wg
wg
```

Informació del problema

Autor : PRO2
Generació : 2024-04-23 16:42:52