Ajustar els salts de ++ i – a la classe List — X57590_ca

Típicament, l'operador ++ dels iteradors de la classe List els desplaça una unitat cap al final de la llista, i l'operador -- dels iteradors de la classe List els desplaça una unitat cap al principi de la llista.
En aquest exercici modificarem i extendrem la classe List. Afegirem dos nous mètodes que permetran modificar com es desplacen els iteradors, en sentit i nombre d'unitats, quan els hi apliquem els operadors ++ i --. Aquests dos nous mètodes s'anomenen `changePlusMove` i `changeMinusMove`, i tots dos reben un iterador `it` per referència i un enter x. El valor absolut de x indica quantes unitats es desplaça `it` quan se li aplica l'operador. A més, si x és positiu, l'iterador s'ha de desplaçar cap al seu sentit habitual (cap al final de la llista en el cas de `changePlusMove` i ++, i cap al principi de la llista en el cas de `changeMinusMove` i --). En canvi, si x és negatiu, llavors el sentit del moviment és l'oposat a l'habitual.
A més a més, modificarem el comportament d'aquests operadors de manera que no produeixen error quan mirem de desplaçar-los més enllà dels límits. En tals casos, simplement no es mouran. Per exemple, si un iterador es troba al `end` de la llista i mirem de desplaçar-lo una o més unitats cap al final, simplement no es mourà, sense produïr error. I si un iterador es troba al principi de la llista i mirem de desplaçar-lo una o més unitats cap al principi, simplement no es mourà, sense produir error.
Fixeu-vos en el següent exemple de programa i el seu comportament descrit en els seus comentaris.

```
List<string> l;                          // l:
l.push_back("a");                        // l: a
l.push_back("b");                        // l: a,b
l.push_back("c");                        // l: a,b,c
l.push_back("d");                        // l: a,b,c,d
l.push_back("e");                        // l: a,b,c,d,e
l.push_back("f");                        // l: a,b,c,d,e,f
l.push_back("g");                        // l: a,b,c,d,e,f,g
l.push_back("h");                        // l: a,b,c,d,e,f,g,h
List<string>::iterator it = l.begin(); // l: (a),b,c,d,e,f,g,h
it++;                                    // l: a,(b),c,d,e,f,g,h
l.changePlusMove(it, 2);
it++;                                    // l: a,b,c,(d),e,f,g,h
it--;                                    // l: a,b,(c),d,e,f,g,h
it++;                                    // l: a,b,c,d,(e),f,g,h
l.changePlusMove(it, 5);
it++;                                    // l: a,b,c,d,e,f,g,h()
it++;                                    // l: a,b,c,d,e,f,g,h()
l.changePlusMove(it, -1);
it++;                                    // l: a,b,c,d,e,f,g,(h)
it++;                                    // l: a,b,c,d,e,f,(g),h
l.changePlusMove(it, -3);
it++;                                    // l: a,b,c,(d),e,f,g,h
it++;                                    // l: (a),b,c,d,e,f,g,h
it++;                                    // l: (a),b,c,d,e,f,g,h
```

```
it--;                                   // l: (a),b,c,d,e,f,g,h
l.changeMinusMove(it, -2);
it--;                                   // l: a,b,(c),d,e,f,g,h
l.changeMinusMove(it, -3);
it--;                                   // l: a,b,c,d,e,(f),g,h
it--;                                   // l: a,b,c,d,e,f,g,h()
it--;                                   // l: a,b,c,d,e,f,g,h()
l.changeMinusMove(it, 1);
it--;                                   // l: a,b,c,d,e,f,g,(h)
it--;                                   // l: a,b,c,d,e,f,(g),h
```

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` les següents línies:

```
// Pre:
// Post: Modifica el comportament de l'operador ++ aplicat a aquest iterador it
//       que a partir d'aquest moment es desplaça x unitats cap al final.
// Descomenteu les següents dues linies i implementeu el mètode:
// void changePlusMove(iterator &it, int x) {
// }

// Pre:
// Post: Modifica el comportament de l'operador -- aplicat a aquest iterador it
//       que a partir d'aquest moment es desplaça x unitats cap al principi.
// Descomenteu les següents dues linies i implementeu el mètode:
// void changeMinusMove(iterator &it, int x) {
// }
```

Descomenteu les linies que s'indiquen i implementeu els mètodes. També caldrà que modifiqueu altres parts convenientment per tal de poder recordar si a un iterador en concret se li ha modificat el seu moviment i com. També haureu d'adaptar els operadors ++ i -- convenientment.
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

L'entrada del programa té una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre la llista i dos iteradors que se suposen situats inicialment al principi (i final) de la llista:

```
push_front s (s és string)
push_back s (s és string)
pop_front
pop_back
it1 = begin
it1 = end
it1 = erase it1
it1++
```

```
it1--
++it1
--it1
*it1 = s (s és string)
insert it1 s (s és string)
cout << *it1
changePlusMove it1 x (x és enter)
changeMinusMove it1 x (x és enter)
it2 = begin
it2 = end
it2 = erase it2
it2++
it2--
++it2
--it2
*it2 = x (x és string)
insert it2 x (x és string)
cout << *it2
changePlusMove it2 x (x és enter)
changeMinusMove it2 x (x és enter)
cout << l
```

Se suposa que la seqüència d'entrada serà correcta, és a dir, que no es produeixen errors d'execució si s'apliquen correctament sobre una llista i dos iteradors amb les condicions abans esmentades.
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 el mètode `sum` abans esmentat.

Sortida

Per a cada instrucció `cout << *it1` o `cout << *it2` s'escriurà el contingut apuntat per l'iterador `it1` o `it2`, respectivament. Per a cada instrucció `cout << l` s'escriurà el contingut de tota la llista. El programa que us oferim ja fa això. Només cal que feu els canvis abans esmentats.

Exemple d'entrada 1

```
cout << l
it1--
--it1
it1++
++it1
it2--
--it2
it2++
++it2
cout << l
push_back a
push_back b
push_back c
cout << l
push_back d
push_back e
push_back f
push_back g
push_back h
cout << l
it1 = begin
cout << l
it1--
--it1
cout << l
it1++
++it1
cout << l
it1 = begin
it2 = end
cout << l
changePlusMove it1 2
```

```
it1++
cout << l
++it1
cout << l
--it1
cout << l
it1--
cout << l
it2++
cout << l
cout << *it1
changeMinusMove it2 2
it2--
cout << l
it2++
cout << l
--it2
cout << l
changePlusMove it2 -2
it2++
cout << l
changePlusMove it2 -3
it1++
cout << l
++it2
it1--
--it1
cout << l
changeMinusMove it2 -2
push_front i
push_back j
push_front k
push_back l
push_front m
cout << l
it1 = end
changePlusMove it1 -2
changeMinusMove it1 2
it1++
it1--
--it1
++it1
it2++
++it2
it2--
--it2
cout << l
it1 = begin
it2 = begin
changePlusMove it1 2
changePlusMove it2 2
changeMinusMove it1 -1
changeMinusMove it2 -3
cout << *it1
cout << *it2
it1--
it1--
it2--
it2--
cout << l
cout << *it1
cout << *it2
insert it1 o
insert it2 p
cout << l
pop_front
pop_back
cout << l
it1--
it2--
cout << l
it1 = erase it1
it2 = erase it2
cout << l
*it1 = x
*it2 = y
cout << l
it1++
it2++
cout << l
it1--
it2--
cout << l
```

Exemple de sortida 1

```
([])
([])
a b c ([])
a b c d e f g h ([])
(a) b c d e f g h []
(a) b c d e f g h []
a b (c) d e f g h []
(a) b c d e f g h []
a b (c) d e f g h []
a b c d (e) f g h []
a b c (d) e f g h []
a b (c) d e f g h []
a b (c) d e f g h []
c
a b (c) d e f [g] h
a b (c) d e f g [h]
```

```
a b (c) d e [f] g h
a b (c) [d] e f g h
a b c [d] (e) f g h
[a] b (c) d e f g h
m k i [a] b (c) d e f g h j l
m k i a [b] (c) d e f g h j l
m
m
m k (i) a b c [d] e f g h j l
i
d
m k o (i) a b c p [d] e f g h j l
k o (i) a b c p [d] e f g h j
k o i (a) b c p d e f [g] h j
k o i (b) c p d e f [h] j
k o i (x) c p d e f [y] j
k o i x c (p) d e f y j []
k o i x c p (d) e f y j []
```

Exemple d'entrada 2

```
push_back hxc
insert it1 v
push_front k
push_front dnx
push_front qy
insert it1 bjo
it1--
changePlusMove it1 1
insert it2 dglx
pop_back
--it2
--it1
it1 = begin
changePlusMove it2 4
insert it1 ec
push_front tj
cout << *it1
it1 = erase it1
push_front esop
changeMinusMove it1 1
it1 = erase it1
*it2 = rpqe
push_front gqd
*it1 = gtb
it2++
push_back e
cout << l
pop_front
changeMinusMove it1 -3
cout << l
push_front knbe
changePlusMove it1 1
pop_back
insert it1 ikrb
push_front fdw
push_front c
cout << l
push_front a
push_back g
changePlusMove it1 -1
```

```
push_front sijp
changeMinusMove it1 1
pop_back
--it1
changePlusMove it1 1
changeMinusMove it2 3
changeMinusMove it2 1
cout << *it1
--it2
*it1 = gfou
changePlusMove it1 -2
cout << l
cout << l
push_back rbr
*it2 = p
it2--
changePlusMove it1 2
*it1 = t
push_front oejy
changeMinusMove it1 1
push_front qms
push_back y
push_front wgtt
push_back dlx
*it1 = qrrg
push_back lqa
*it1 = ho
push_front a
changeMinusMove it2 -4
push_back fju
push_back kx
*it2 = fndr
insert it2 ni
cout << *it2
push_back seb
cout << l
it2++
changeMinusMove it1 -4
cout << l
push_front pvji
insert it1 rozw
```

```
push_back asjy
cout << *it1
it2 = erase it2
push_back nt
push_front k
++it2
*it1 = mqai
changePlusMove it1 3
--it2
push_back ybe
changePlusMove it1 2
pop_front
pop_front
push_front ssdi
push_back e
pop_back
insert it2 kfci
pop_back
cout << *it1
*it1 = za
cout << *it1
push_front knwi
push_back hcdr
insert it1 lpc
it1 = erase it1
*it1 = f
push_front ul
changeMinusMove it1 0
push_front ei
insert it2 k
insert it1 g
it1--
push_back vh
it2++
changePlusMove it2 4
cout << l
push_front gmbn
changeMinusMove it2 0
*it1 = s
push_back xwi
cout << l
*it1 = z
push_back ogou
insert it2 amgm
changeMinusMove it1 -3
it2--
push_back x
insert it2 xgg
changeMinusMove it1 -3
insert it1 ya
push_front ojx
push_front lf
push_back exzm
push_back ibxw
--it2
push_back kru
pop_front
*it1 = e
insert it2 y
push_front m
push_back fp
push_back q
changeMinusMove it1 0
push_back h
push_back qgs
push_back algr
push_front bf
changePlusMove it2 0
push_front ey
push_back a
changeMinusMove it1 1
push_back ios
push_back ssdu
it1 = erase it1
push_back evm
changePlusMove it2 -1
push_front qxz
changePlusMove it2 1
push_back pb
cout << l
push_front gz
push_back uz
*it1 = z
push_back cl
push_front iuyj
cout << l
push_back fj
changeMinusMove it1 0
changePlusMove it2 -3
cout << *it1
changeMinusMove it1 4
push_front tw
push_back kva
++it1
push_front n
cout << l
insert it2 uyv
cout << l
```

Exemple de sortida 2

```
qy
gqd esop tj ec (gtb) hxc v rpqe e []
esop tj ec (gtb) hxc v rpqe e []
c fdw knbe esop tj ec ikrb (gtb) hxc v rpqe []
ikrb
sijp a c fdw knbe esop tj ec (gfou) gtb hxc v [rpqe]
sijp a c fdw knbe esop tj ec (gfou) gtb hxc v [rpqe]
fndr
a wgtt qms oejy sijp a c fdw knbe esop tj ec (ho) gtb hxc nj [fndr] p rbn y dlx lqa fju kx seb
```

```
a wgtt qms oejy sijp a c fdw knbe esop tj ec (ho) gtb h
ho
mqai
za
ei ul knwi ssdi a wgtt qms oejy sijp a c fdw knbe esop
gmbn ei ul knwi ssdi a wgtt qms oejy sijp a c fdw knbe
qxz ey bf m ojx gmbn ei ul knwi ssdi a wgtt qms oejy si
iuyj gz qxz ey bf m ojx gmbn ei ul knwi ssdi a wgtt qms
z
n tw iuyj gz qxz ey bf m ojx gmbn ei ul knwi ssdi a wgt
n tw iuyj gz qxz ey bf m ojx gmbn ei ul knwi ssdi a wgt
```

Observació

Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, on cada operació té cost constant o proporcional al nombre de desplaçaments necessaris a aplicar sobre els operadors (en els casos ++ i --), i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autoria: PRO2

Generació: 2026-01-27T18:53:29.905Z