
Suma d'una llista

X65481_ca

Preliminars

En aquest exercici extendrem la classe `List` suposant que el tipus `T` dels elements de la llista té definida la operació de suma `+`, és a dir, que dues variables de tipus `T` es poden sumar. Aquesta suma satisfarà la propietat d'associativitat usual ($x + (y + z) = (x + y) + z$), i tindrà element neutre (per exemple, `0` és el neutre per a la suma d'enters, i l'string buit és el neutre per a la suma d'strings).

També suposem que una variable `x` de tipus `T` permet una assignació `x = 0`, de manera que, internament, a `x` se li assigna el neutre de la suma.

Testejarem l'exercici amb el tipus `int` i amb un tipus contenidor d'string que permetrà fer assignacions `x = 0` (internament s'assignarà l'string buit). Fixeu-vos que en el cas d'strings la suma és de fet la concatenació d'strings, i que no és commutativa. Per exemple, `"a"+"b" = "ab" ≠ "ba" = "b"+"a"`.

Exercici

Implementeu un nou mètode de la classe `List` que retorni la suma dels elements continguts a la llista des del principi fins al final. És a dir, si $[a_1, \dots, a_n]$ és el contingut de la llista, el mètode ha de retornar $a_1 + \dots + a_n$.

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: Sigui [a1, ..., an] el contingut actual de la llista.
// Post: Retorna a1+...+an.
// Descomenteu les següents dues línies i implementeu el mètode:
// T sum() {
// }
```

Descomenteu les dues línies que s'indiquen i implementeu el 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.

Observació: Una implementació senzilla inicialitzant una variable de tipus `T` a `0` (que es transformarà automàticament en el neutre de la suma) i recorrent i sumant-hi els elements de la llista hauria de ser suficient per a superar tots els jocs de proves.

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 primera línia amb o bé `int` o bé `string`, que indica el tipus `T` dels elements de la llista `l` amb la que treballarà el programa, que se suposa inicialment buida. Després, hi ha una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre la llista i un iterador que se suposa situat inicialment al principi (i final) de la llista:

```
push_front x (x és de tipus T)
push_back x (x és de tipus T)
pop_front
pop_back
```

```

it = begin
it = end
it = erase it
it++
it--
++it
--it
*it = x (x és de tipus T)
insert it x (x és de tipus T)
cout << *it
cout << sum
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 un iterador 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 << *it` s'escriurà el contingut apuntat per l'iterador. Per a cada instrucció `cout << l` s'escriurà el contingut de tota la llista. Per a cada instrucció `cout << sum` s'escriurà la suma de la llista. El programa que us oferim ja fa això. Només cal que implementeu el mètode `sum` abans esmentat.

Exemple d'entrada 1

```

int
cout << l
cout << sum
push_back 1
push_back 2
push_back 3
cout << l
cout << sum
it = begin
insert it -1
--it
push_front 4
push_front -2
cout << l
cout << sum
*it = 5
cout << l
cout << sum
it++
it = erase it
cout << l
cout << sum
it = end
cout << l
cout << sum
--it

```

```

it = erase it
push_back 6
cout << l
cout << sum
it--
--it
*it = 0
cout << l
cout << sum
pop_back
push_back 7
cout << l
cout << sum
insert it 8
cout << l
cout << sum
++it
pop_front
it = erase it
cout << l
cout << sum
insert it 9
cout << l
cout << sum

```

Exemple de sortida 1

```
()
0
1 2 3 ()
6
-2 4 (-1) 1 2 3
7
-2 4 (5) 1 2 3
13
-2 4 5 (2) 3
12
-2 4 5 2 3 ()
```

Exemple d'entrada 2

```
string
cout << l
cout << sum
push_back a
push_back b
push_back c
cout << l
cout << sum
it = begin
insert it dd
--it
push_front ee
push_front f
cout << l
cout << sum
*it = g
cout << l
cout << sum
it++
it = erase it
cout << l
cout << sum
it = end
cout << l
cout << sum
--it
it = erase it
push_back hhh
cout << l
cout << sum
it--
--it
*it = iiii
cout << l
cout << sum
pop_back
push_back j
cout << l
cout << sum
insert it kkkk
cout << l
cout << sum
++it
pop_front
it = erase it
```

```
12
-2 4 5 2 6 ()
15
-2 4 5 (0) 6
13
-2 4 5 (0) 7
14
-2 4 5 8 (0) 7
22
4 5 8 0 ()
17
4 5 8 0 9 ()
26
```

```
cout << l
cout << sum
insert it mmm
cout << l
cout << sum
```

Exemple de sortida 2

```
()  
  
a b c ()  
abc  
f ee (dd) a b c  
feeddabc  
f ee (g) a b c  
feegabc  
f ee g (b) c  
feegbc  
f ee g b c ()
```

Exemple d'entrada 3

```
int  
push_back 8  
pop_back  
push_front 10  
push_back 1  
push_back 7  
it = begin  
pop_back  
cout << sum  
push_front 10  
push_back 7  
*it = 3  
pop_back  
cout << sum  
insert it 8  
cout << sum  
cout << sum  
push_front 4  
cout << sum  
insert it 7  
insert it 1  
cout << *it  
pop_back  
cout << *it  
insert it 7  
push_back 8  
it++  
insert it 8  
insert it 4  
push_back 9  
insert it 9  
push_front 2  
push_back 3  
cout << *it  
*it = 9  
*it = 3  
*it = 7  
cout << *it  
cout << *it  
cout << *it  
it = erase it  
cout << *it  
push_front 8  
cout << sum  
cout << l
```

```
feegbc  
f ee g b hhh ()  
feegbhhh  
f ee g (iiii) hhh  
feegiiiihhh  
f ee g (iiii) j  
feegiiiiij  
f ee g kkkk (iiii) j  
feegkkkkiiiiij  
ee g kkkk iiii ()  
eegkkkkiiii  
ee g kkkk iiii mmm ()  
eegkkkkiiiiimmm
```

Exemple de sortida 3

```
11  
14  
22  
22  
26  
3  
3  
8  
7  
7  
7  
9  
83  
8 2 4 10 8 7 1 7 3 8 4 9 (9) 3
```

Exemple d'entrada 4

```
string
push_back rb
push_front hc
push_back k
push_back iddq
pop_back
cout << sum
push_front rx
push_back bld
insert it sa
push_front yn
insert it yg
pop_back
insert it lln
insert it wkho
cout << sum
insert it q
push_back u
it--
*it = gbb
it = erase it
push_front iv
push_back d
insert it bx
insert it v
cout << l
--it
push_front nsnf
insert it fjm
*it = rr
*it = o
push_back cn
pop_back
push_front fb
cout << sum
*it = x
cout << sum
cout << l
```

Exemple de sortida 4

```
hcrbk
ynrxhcrbkbldsallnwkho
iv yn rx hc rb k bld sa lln wkho q d bx v ()
fbnsnfivynrxhcrbkbldsallnwkhoqdbxfjmo
fbnsnfivynrxhcrbkbldsallnwkhoqdbxfjmx
fb nsnf iv yn rx hc rb k bld sa lln wkho q d bx fjm (x)
```

Observació

Avaluació sobre 10 punts: (Afegiu comentaris si el vostre codi no és prou clar)

- Solució lenta: 6 punts.
- solució ràpida: 10 punts.

Entenem com a solució lenta una que és correcta i capaç de superar els jocs de proves públics. Entenem com a solució ràpida una que és correcta i capaç de superar els jocs de proves públics i privats.

Informació del problema

Autor : PRO2

Generació : 2024-04-23 16:44:39

© *Jutge.org*, 2006–2024.
<https://jutge.org>