Iteradors quàntics a la classe List — X66290_ca

Típicament, executar `++` sobre un iterador que es troba al end de la llista produeix error d'execució, i executar `--` sobre un iterador que es troba al begin de la llista també produeix error d'execució. Per començar, en aquest exercici modificarem la subclasse `iterator` de la classe List de manera que els errors d'execució abans esmentats ja no es produiran. Simplement, en tals casos els iteradors no es mouran.
Després modificarem la classe `iterator` afegint dos nous mètodes `entangle` i `disentangle` i canviant el comportament dels mètodes `++` i `--` com descrivim a continuació.
El nou mètode `entangle` rebrà un altre `iterator` com a paràmetre (és a dir, un iterador del mateix tipus, tot i que potser apunta a un element d'una llista diferent). Una crida `it0.entangle(it1)` provocarà que `it0` quedi enllaçat a `it1` de manera que, a partir de llavors, sempre que executem `it0++` o `++it0`, l'iterador `it1` també es desplaçarà cap al final de la seva llista corresponent (si no es troba ja al end). A més a més, sempre que executem `it0--` o `--it0`, l'iterador `it1` també es desplaçarà cap al principi de la seva llista corresponent (si no es troba ja al begin).
Aquest efecte no es propaga per una seqüència d'enllaços. Per exemple, si hem executat `it0.entangle(it1)` i `it1.entangle(it2)`, executar `it0++` mourà també `it1` cap al final de la seva llista, però això no es propagarà a moure `it2` cap al final de la seva llista.
Successius `entangle` sobre un mateix iterador fan que només l'últim estigui actiu. Per exemple, si hem executat `it0.entangle(it1)` i després `it0.entangle(it2)`, llavors `it0` està enllaçat a `it2` però no a `it1`.
Una crida `it0.disentangle()` cancel.larà l'efecte de l'últim `entangle` actiu.
Fixeu-vos en aquest exemple per tal d'acabar d'entendre-ho:

```
List<int> l0, l1;
List<int>::iterator a, b, c, d;

l0.push_back(1);     // l0: 1,
l0.push_back(2);     // l0: 1,2,
l0.push_back(3);     // l0: 1,2,3,
l1.push_back(4);     // l1: 4,
l1.push_back(5);     // l1: 4,5,
l1.push_back(6);     // l1: 4,5,6,

a = l0.begin();      // l0: 1a,2,3,
b = l0.end();        // l0: 1a,2,3,b
c = l1.begin();      // l1: 4c,5,6,
d = l1.begin();      // l1: 4c,5,6,d

a--;                 // l0: 1a,2,3,b
a++;                 // l0: 1,2a,3,b
b++;                 // l0: 1,2a,3,b
b--;                 // l0: 1,2a,3b,
a.entangle(b);
a++;                 // l0: 1,2,3a,b
a++;                 // l0: 1,2,3,ab
```

```
a++;                  // l0: 1,2,3,ab
a--;                  // l0: 1,2,3ab,
b--;                  // l0: 1,2b,3a,
a--;                  // l0: 1b,2a,3,
a--;                  // l0: 1ab,2,3,
a--;                  // l0: 1ab,2,3,

a.entangle(c);
c.entangle(d);
a++;                  // l0: 1b,2a,3,  l1: 4,5c,6,d
c++;                  // l0: 1b,2a,3,  l1: 4,5,6c,d
c--;                  // l0: 1b,2a,3,  l1: 4,5c,6d,
c--;                  // l0: 1b,2a,3,  l1: 4c,5d,6,
a++;                  // l0: 1b,2,3a,  l1: 4,5cd,6,
a++;                  // l0: 1b,2,3,a  l1: 4,5d,6c,
a.disentangle();
a++;                  // l0: 1b,2,3,a  l1: 4,5d,6c,
a--;                  // l0: 1b,2,3a,  l1: 4,5d,6c,
c++;                  // l0: 1b,2,3a,  l1: 4,5,6d,c
c.disentangle();
c++;                  // l0: 1b,2,3a,  l1: 4,5,6d,c
c--;                  // l0: 1b,2,3a,  l1: 4,5,6cd,
```

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 d'implementar els dos nous mètodes `entangle` i `disentangle` dins `list.hh` a la part pública de la classe `iterator` (podeu trobar les capçaleres comentades dins `list.hh`), i modificar els dos mètodes `++` i els dos mètodes `--` convenientment (en realitat només cal modificar el pre-increment i el pre-decrement perquè el post-increment i post-decrement criden als primers). Necessitareu també algun atribut addicional per tal de recordar si l'iterador té un `entangle` actiu i amb qui, amb les convenients inicialitzacions.
Més concretament, heu de fer els canvis que s'indiquen en algunes parts del codi de list.hh:

```
// Iterators mutables
class iterator {
  friend class List;
  private:
    List *plist;
    Item *pitem;
    // Add what it takes to know if the iterator is entangled to another iter

public:

  iterator() {
    // Add initialization of new attributes.
  }

  public:

  // Adapt this function so that moving beyond boundaries does not trigger er
```

```
    // but leaves the iterator unchanged instead.
    // Also, add the necessary adaptations so that, the (possible) entangled it
    // moves accordingly.
    // Preincrement
    iterator operator++()
    /* Pre: el p.i apunta a un element E de la llista,
       que no és el end() */
    /* Post: el p.i apunta a l'element següent a E
       el resultat és el p.i. */
    {
      if (pitem == &(plist->itemsup)) {
        cerr << "Error: ++iterator at the end of list" << endl;
        exit(1);
      }
      pitem = pitem->next;
      return *this;
    }

...

    // Adapt this function so that moving beyond boundaries does not trigger er
    // but leaves the iterator unchanged instead.
    // Also, add the necessary adaptations so that, the (possible) entangled it
    // moves accordingly.
    // Predecrement
    iterator operator--()
    /* Pre: el p.i apunta a un element E de la llista que
       no és el begin() */
    /* Post: el p.i apunta a l'element anterior a E,
       el resultat és el p.i. */
    {
      if (pitem == plist->iteminf.next) {
       cerr << "Error: --iterator at the beginning of list" << endl;
       exit(1);
      }
      pitem = pitem->prev;
      return *this;
    }

...

    // Pre: 'it' != 'this'
    // Post: Once executed, any move (++ or --) on 'this' will affect 'it' the
    //       Any previous entangle or disentangle is cancelled.
    // Remove comment marks and implement this function:
    // void entangle(iterator &it) {
    // }

    // Pre: 'this' is entangled
```

```
    // Post: Previous entangle is cancelled.
    // Remove comment marks and implement this function:
    // void disentangle() {
    // }

...
```

No cal decidir que passa amb assignacions entre iteradors existents, doncs no es consideraran en 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 comença amb una declaració d'unes quantes llistes (`l0, l1, ...`) i uns quants iteradors (`a,b,c,...`), i després té una seqüència de comandes sobre les llistes i els iteradors declarats. Com que ja us oferim el `main.cc`, no cal que us preocupeu d'implementar la lectura d'aquestes entrades. Només cal que implementeu la extensió de la classe `iterator` abans esmentada.
Per simplificar, no hi haurà comandes que eliminin elements de les llistes, com `pop_back`, `pop_front i erase`. Podeu suposar que les comandes no fan coses extranyes, com fer que un iterador estigui enllaçat per entangle a si mateix, i que sempre que un iterador sigui mogut (directament o indirectament per entangle), aquest estarà apuntant a alguna posició d'alguna llista. Podeu suposar que les comandes faran disentangles només sobre iteradors que tinguin un entangle actiu. Però pot ser el cas que es faci un entangle sobre una iterador que ja tingui un entangle actiu. Com mencionavem abans, en aquestes situacions només l'últim entangle aplica.

Sortida

Per a cada comanda d'escriptura sobre la sortida s'escriurà el resultat corresponent. El `main.cc` que us oferim ja fa això. Només cal que implementeu la extensió de la classe `iterator` abans esmentada.

Exemple d'entrada 1

```
List<int> l0 , l1 ;
List<int>::iterator a , b , c , d ;
l0 .push_back( 1 );
l0 .push_back( 2 );
l0 .push_back( 3 );
l1 .push_back( 4 );
l1 .push_back( 5 );
l1 .push_back( 6 );
a = l0 .begin();
b = l0 .end();
c = l1 .begin();
d = l1 .end();
cout<< l0 <<endl;
cout<< l1 <<endl;
a --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
b ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
b --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a .entangle( b );
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
a ++;
cout<< l0 <<endl;
```

```
cout<< l1 <<endl;
a --;
cout<< l0 <<endl;
cout<< l1 <<endl;
b --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a .entangle( c );
c .entangle( d );
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
c ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
c --;
cout<< l0 <<endl;
cout<< l1 <<endl;
c --;
cout<< l0 <<endl;
cout<< l1 <<endl;
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
a .disentangle();
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
a --;
cout<< l0 <<endl;
cout<< l1 <<endl;
c ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
c .disentangle();
c ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
c --;
cout<< l0 <<endl;
cout<< l1 <<endl;
```

Exemple de sortida 1

```
1a,2,3,b
4c,5,6,d
1a,2,3,b
4c,5,6,d
1,2a,3,b
4c,5,6,d
1,2a,3,b
4c,5,6,d
1,2a,3b,
4c,5,6,d
1,2,3a,b
4c,5,6,d
1,2,3,ab
4c,5,6,d
1,2,3,ab
4c,5,6,d
1,2,3ab,
4c,5,6,d
1,2b,3a,
4c,5,6,d
1b,2a,3,
4c,5,6,d
1ab,2,3,
4c,5,6,d
1ab,2,3,
4c,5,6,d
1b,2a,3,
4,5c,6,d
1b,2a,3,
4,5,6c,d
1b,2a,3,
4,5c,6d,
1b,2a,3,
4c,5d,6,
1b,2,3a,
4,5cd,6,
1b,2,3,a
4,5d,6c,
1b,2,3,a
4,5d,6c,
1b,2,3a,
4,5d,6c,
1b,2,3a,
4,5,6d,c
1b,2,3a,
4,5,6d,c
1b,2,3a,
4,5,6cd,
```

Exemple d'entrada 2

```
List<int> l0 , l1 , l2 ;
List<int>::iterator a , b , c ;
a = l1 .begin();
b = l1 .begin();
c = l0 .begin();
l2 .push_front( 3 );
-- a ;
cout<< l1 <<endl;
b .entangle( a );
++ b ;
c ++;
```

```
cout<< l1 <<endl;
l0 .insert( c , -4 );
-- a ;
l1 .push_back( 0 );
cout<< l0 <<endl;
a .entangle( c );
b ++;
l0 .push_front( -2 );
c --;
b .entangle( a );
++ a ;
l0 .insert( c , 3 );
b = l2 .end();
cout<< l0 <<endl;
++ a ;
l2 .push_back( 2 );
l1 .push_back( 2 );
l2 .push_front( 1 );
a --;
l0 .push_front( 2 );
l0 .push_front( -3 );
b .entangle( c );
c .entangle( b );
c .entangle( a );
a --;
l0 .push_front( 3 );
l1 .push_front( -2 );
++ c ;
a = l0 .begin();
l0 .push_front( -2 );
cout<< l2 <<endl;
b --;
l0 .insert( a , 2 );
cout<< l1 <<endl;
cout<< l2 <<endl;
b --;
a --;
b .disentangle();
l1 .push_front( -1 );
l2 .push_front( 1 );
cout<< l2 <<endl;
b .entangle( a );
l2 .insert( b , 4 );
++ b ;
c ++;
l2 .push_back( 4 );
-- a ;
l2 .insert( b , 0 );
l2 .push_back( -3 );
l2 .push_front( -1 );
cout<< l2 <<endl;
l2 .push_front( -3 );
l1 .push_front( 1 );
a --;
c = l1 .end();
cout<< l1 <<endl;
c .entangle( a );
b = l1 .begin();
a = l2 .end();
l2 .push_front( -1 );
l2 .push_back( 4 );
l1 .insert( b , -2 );
cout<< l0 <<endl;
c ++;
c ++;
l0 .push_front( -4 );
cout<< l2 <<endl;
c .entangle( a );
l2 .insert( a , 0 );
++ c ;
cout<< l1 <<endl;
b .entangle( a );
c = l1 .begin();
a --;
l2 .insert( a , -2 );
cout<< l0 <<endl;
c ++;
c .entangle( b );
l2 .push_back( 4 );
c .entangle( b );
b .entangle( c );
-- a ;
l1 .insert( b , -1 );
b .entangle( c );
l1 .push_back( -2 );
c ++;
cout<< l0 <<endl;
-- a ;
l1 .push_front( -1 );
cout<< l2 .size()<<endl;
-- b ;
l1 .insert( c , -1 );
-- a ;
b .entangle( a );
l1 .insert( b , -4 );
b --;
++ c ;
l0 .push_back( 0 );
b = l1 .begin();
c --;
cout<< l1 <<endl;
c .entangle( a );
++ a ;
c = l2 .begin();
l2 .push_back( -2 );
cout<< l2 <<endl;
a = l0 .end();
l0 .insert( a , -3 );
l1 .push_back( 4 );
a = l2 .end();
b .entangle( c );
++ c ;
cout<< l2 <<endl;
cout<< l2 .size()<<endl;
b .entangle( c );
b ++;
b ++;
b .disentangle();
l0 .push_back( 2 );
l2 .insert( a , -1 );
l2 .push_back( 2 );
b .entangle( c );
```

```
a .entangle( c );
c --;
l2 .push_front( -3 );
-- b ;
b .entangle( c );
cout<< l2 <<endl;
l1 .push_front( -2 );
l0 .push_front( 0 );
c --;
c --;
b ++;
l1 .insert( b , -3 );
l2 .push_front( -1 );
cout<< l2 <<endl;
cout<< l2 <<endl;
l1 .insert( b , 1 );
l2 .insert( c , 3 );
a .entangle( c );
c --;
cout<< l1 <<endl;
b --;
l2 .push_front( 4 );
c = l1 .end();
c .entangle( b );
a ++;
a --;
l2 .push_back( -4 );
a = l0 .begin();
cout<< l0 <<endl;
a --;
cout<< l2 .size()<<endl;
l0 .push_back( 0 );
cout<< l2 <<endl;
c .entangle( b );
l2 .push_front( 0 );
a ++;
l2 .push_front( -4 );
a = l1 .end();
cout<< l1 .size()<<endl;
a .entangle( c );
l1 .push_back( -3 );
l1 .push_back( 2 );
l2 .push_front( 1 );
a .entangle( b );
a = l2 .end();
l2 .push_front( 1 );
l0 .push_back( 3 );
l1 .insert( b , 3 );
cout<< l2 <<endl;
l2 .insert( a , 3 );
c = l0 .end();
l0 .insert( c , -4 );
a --;
l2 .push_back( -4 );
cout<< l1 <<endl;
l0 .push_front( 0 );
l0 .push_back( -4 );
l0 .insert( c , 3 );
++ a ;
l1 .push_front( -4 );
l1 .push_front( 3 );
l1 .insert( b , 1 );
cout<< l1 <<endl;
l1 .push_back( 0 );
l1 .push_back( 2 );
l2 .push_back( 3 );
l2 .insert( a , 1 );
l1 .push_front( 4 );
-- b ;
l0 .insert( c , -4 );
l2 .push_back( 4 );
cout<< l1 <<endl;
cout<< l0 <<endl;
cout<< l1 <<endl;
cout<< l2 <<endl;
```

Exemple de sortida 2

```
ab
ab
-4,c
-2,-4,3,c
1,3,2,b
-2,0,2,
1,3,2b,
1,1,3b,2,
-1,1,1,4,3,0,2b,4,-3,
1,-1,-2,0,2,c
-2,2,3,-3,2,-2,-4,3,
-1,-3,-1,1,1,4,3,0,2,4,-3,4,a
-2,1b,-1,-2,0,2,c
-4,-2,2,3,-3,2,-2,-4,3,
-4,-2,2,3,-3,2,-2,-4,3,
15
-1b,-2,-1,-1,-4,1c,-1,-2,0,2,-2,
-1c,-3,-1,1,1,4,3,0,2,4,-3a,4,-2,0,4,-2,
-1,-3c,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,-2,a
16
-3,-1,-3c,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,-2,-1,2,a
-1,-3,-1c,-3,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,-2,-1,2,a
-1,-3,-1c,-3,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,-2,-1,2,a
-2,-1,-2,-3,1,-1b,-1,-4,1,-1,-2,0,2,-2,4,
0a,-4,-2,2,3,-3,2,-2,-4,3,0,-3,2,
23
4,-1,-3,3,-1,-3,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,-2,-1,2,-
15
1,1,-4,0,4,-1,-3,3,-1,-3,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,
-2,-1,-2,-3,3,1b,-1,-1,-4,1,-1,-2,0,2,-2,4,-3,2,
3,-4,-2,-1,-2,-3,3,1,1b,-1,-1,-4,1,-1,-2,0,2,-2,4,-3,2
4,3,-4,-2,-1,-2,-3,3,1b,1,-1,-1,-4,1,-1,-2,0,2,-2,4,-3
0,0,-4,-2,2,3,-3,2,-2,-4,3,0,-3,2,0,3,-4,-4,-4,3c,
4,3,-4,-2,-1,-2,-3,3,1b,1,-1,-1,-4,1,-1,-2,0,2,-2,4,-3
1,1,-4,0,4,-1,-3,3,-1,-3,-1,1,1,4,3,0,2,4,-3,4,-2,0,4,
```

Observació

Avaluació sobre 10 punts:

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

Entenem com a solució ràpida una que és correcta, on totes les operacions tenen cost constant (excepte l'escriptura de tota la llista per la sortida, que té cost lineal), 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

Autor : PRO2
Generació : 2024-04-24 16:04:00