Evitar a un altre iterador a base de quedar-se quiet — X20209_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 `avoid` i `stopAvoid`, i canviant el comportament dels mètodes `++` i `--` com descrivim a continuació.
El nou mètode `avoid` 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.avoid(it1)` provocarà que, a partir d'ara, `it0` intenti evitar apuntar al mateix lloc que `it1`, a base d'evitar moviments que ho poden provocar.
Més concretament, amb una crida `it0++` o `++it0`, l'iterador `it0` no es mourà si fer-ho provoca que `it0` apunti al mateix lloc que `it1`. En particular, si `it0` apunta a l'últim element de la llista i `it1` apunta al `end` de la llista, llavors les crides `it0++` o `++it0` no provocaran cap canvi.
Anàlogament, amb una crida `it0--` o `--it0`, l'iterador `it0` no es mourà si fer-ho provoca que `it0` apunti al mateix lloc que `it1`. En particular, si `it0` apunta al segon element de la llista i `it1` apunta al primer element de la llista, llavors les crides `it0--` o `--it0` no provocaran cap canvi.
Fixeu-vos que la crida `it0.avoid(it1)` no imposa restriccions al moviment de `it1`. Per tant, a base de fer crides que mouen `it1`, pot acabar passant que `it0` i `it1` apuntint al mateix lloc.
Una crida posterior `it0.avoid(it2)` posa restriccions al moviment de `it0` respecte de `it2`, però també deixa sense efecte la crida anterior `it0.avoid(it1)`, és a dir, cancel.la les restriccions del moviment de `it0` respecte de `it1`.
Una crida posterior `it0.stopAvoid()` cancel.la les restriccions del moviment de `it0` respecte de qualsevol altre iterador.
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.end();        // 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.avoid(b);
a++;                  // l0: 1,2a,3b,
b--;                  // l0: 1,2ab,3,
a++;                  // l0: 1,2b,3a,
a--;                  // l0: 1,2b,3a,
b++;                  // l0: 1,2,3ab,
a.avoid(c);
c.avoid(d);
d.avoid(c);
a++;                  // l0: 1,2,3b,a  l1: 4c,5,6,d
a--;                  // l0: 1,2,3ab,  l1: 4c,5,6,d
c--;                  // l0: 1,2,3ab,  l1: 4c,5,6,d
c++;                  // l0: 1,2,3ab,  l1: 4,5c,6,d
c++;                  // l0: 1,2,3ab,  l1: 4,5,6c,d
c++;                  // l0: 1,2,3ab,  l1: 4,5,6c,d
d--;                  // l0: 1,2,3ab,  l1: 4,5,6c,d
c--;                  // l0: 1,2,3ab,  l1: 4,5c,6,d
d--;                  // l0: 1,2,3ab,  l1: 4,5c,6d,
c.stopAvoid();
c++;                  // l0: 1,2,3ab,  l1: 4,5,6cd,
c++;                  // l0: 1,2,3ab,  l1: 4,5,6d,c
d++;                  // l0: 1,2,3ab,  l1: 4,5,6d,c
d.stopAvoid();
d++;                  // l0: 1,2,3ab,  l1: 4,5,6,cd
```

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 `avoid` i `stopAvoid` 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 `avoid` 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 new attributes to remember if the iterator has an active 'avoid'
    // and with which other iterator.

  public:

  iterator() {
```

```
      // Add initialization of new attributes.
    }


    // 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 move does not take plac
    // when there is an active 'avoid' and such a move implies pointing to the
    // the other involved iterator
    // 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 move does not take plac
    // when there is an active 'avoid' and such a move implies pointing to the
    // the other involved iterator
    // 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 attempt (++ or --) on 'this' will cause no
    //       if such a move makes 'this' point to the same place as 'it'.
    //       All former avoid's are cancelled.
    // Remove comment marks and implement this function:
    // void avoid(iterator &it) {
    // }

    // Pre: 'this' has an active avoid.
    // Post: All former avoid's are cancelled.
    // Remove comment marks and implement this function:
    // void stopAvoid() {
    // }

...
```

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 tingui un avoid a si mateix, i que sempre que un iterador sigui mogut, aquest estarà apuntant a alguna posició d'alguna llista. Podeu suposar que les comandes faran `stopAvoid` només sobre iteradors que tinguin un avoid actiu. Però pot ser el cas que es faci un `avoid` sobre un iterador que ja tingui un `avoid` actiu. Com mencionavem abans, en aquestes situacions només l'últim `avoid` 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: 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 .end();          // l1: 4c,5,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;
```

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

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                 // l0: 1,2a,3,b

cout<< l0 <<endl;
cout<< l1 <<endl;

b ++;                 // l0: 1,2a,3,b

cout<< l0 <<endl;
cout<< l1 <<endl;

b --;                 // l0: 1,2a,3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

a .avoid( b );
a ++;                 // l0: 1,2a,3b,

cout<< l0 <<endl;
cout<< l1 <<endl;

b --;                 // l0: 1,2ab,3,

cout<< l0 <<endl;
cout<< l1 <<endl;

a ++;                 // l0: 1,2b,3a,

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                 // l0: 1,2b,3a,

cout<< l0 <<endl;
cout<< l1 <<endl;

b ++;                 // l0: 1,2,3ab,

cout<< l0 <<endl;
cout<< l1 <<endl;

a .avoid( c );
c .avoid( d );
d .avoid( c );
a ++;                 // l0: 1,2,3b,a  l1:c,4c,5,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

a --;                 // l0: 1,2,3ab,  l1: 4c,5,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

c --;                 // l0: 1,2,3ab,  l1: 4c,5,6,d
```

```
cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                 // l0: 1,2,3ab,  l1: 4,5c,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                 // l0: 1,2,3ab,  l1: 4,5,6c,d

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                 // l0: 1,2,3ab,  l1: 4,5,6c,d

cout<< l0 <<endl;
cout<< l1 <<endl;

d --;                 // l0: 1,2,3ab,  l1: 4,5,6c,d

cout<< l0 <<endl;
cout<< l1 <<endl;

c --;                 // l0: 1,2,3ab,  l1: 4,5c,6,d

cout<< l0 <<endl;
cout<< l1 <<endl;

d --;                 // l0: 1,2,3ab,  l1: 4,5c,6d,

cout<< l0 <<endl;
cout<< l1 <<endl;

c .stopAvoid();
c ++;                 // l0: 1,2,3ab,  l1: 4,5,6cd,

cout<< l0 <<endl;
cout<< l1 <<endl;

c ++;                 // l0: 1,2,3ab,  l1: 4,5,6d,c
d ++;                 // l0: 1,2,3ab,  l1: 4,5,6d,c

cout<< l0 <<endl;
cout<< l1 <<endl;

d .stopAvoid();
d ++;                 // l0: 1,2,3ab,  l1: 4,5,6,cd

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

Exemple d'entrada 2

```
List<int> l0 , l1 ;
List<int>::iterator a , b , c , d , e ;
a = l1 .begin();
b = l0 .begin();
c = l1 .begin();
d = l1 .begin();
e = l1 .begin();
b .avoid( c );
b .avoid( e );
a = l1 .begin();
b .stopAvoid();
b ++;
cout<< l0 <<endl;
e = l0 .begin();
-- c ;
e ++;
-- e ;
++ b ;
b = l1 .end();
cout<< l1 <<endl;
e --;
b .avoid( a );
cout<< l0 <<endl;
c = l0 .begin();
cout<< l1 <<endl;
l1 .push_back( 1 );
b = l1 .end();
cout<< l1 <<endl;
c = l0 .end();
c --;
cout<< l0 <<endl;
cout<< l1 .size()<<endl;
c ++;
d .avoid( e );
cout<< l1 <<endl;
e ++;
cout<< l0 <<endl;
cout<< l0 <<endl;
++ b ;
a .avoid( d );
c .avoid( e );
c ++;
a = l0 .begin();
a --;
l0 .push_back( 2 );
-- c ;
a = l1 .end();
++ c ;
cout<< l0 <<endl;
cout<< l1 <<endl;
l1 .insert( d , -1 );
e .avoid( d );
a ++;
cout<< l0 <<endl;
b .avoid( a );
++ a ;
cout<< l1 <<endl;
d .avoid( e );
++ b ;
e ++;
cout<< l1 <<endl;
d .avoid( c );
a --;
l1 .insert( b , 4 );
a --;
-- d ;
a --;
cout<< l1 <<endl;
d .stopAvoid();
l0 .push_back( -3 );
d .avoid( a );
```

```
cout<< l0 .size()<<endl;
e .avoid( c );
-- d ;
++ b ;
cout<< l1 <<endl;
++ e ;
a .avoid( c );
l0 .push_back( 3 );
c ++;
c --;
cout<< l1 <<endl;
cout<< l0 <<endl;
cout<< l1 <<endl;
d .avoid( e );
++ b ;
b = l0 .begin();
cout<< l1 .size()<<endl;
b ++;
a ++;
++ d ;
e ++;
e --;
-- b ;
a = l1 .end();
cout<< l0 <<endl;
-- e ;
-- e ;
a --;
e ++;
l0 .insert( b , 2 );
c .avoid( e );
++ e ;
++ d ;
cout<< l0 <<endl;
c .avoid( b );
b --;
l1 .push_back( 4 );
b = l0 .begin();
l1 .push_back( -2 );
++ a ;
e ++;
cout<< l0 <<endl;
b = l1 .end();
e ++;
a .avoid( c );
l0 .push_back( 3 );
++ c ;
l1 .insert( a , -1 );
e .avoid( b );
++ b ;
cout<< l0 <<endl;
b = l0 .end();
cout<< l0 <<endl;
++ e ;
cout<< l1 <<endl;
-- e ;
b --;
c ++;
c ++;
c = l0 .end();
cout<< l0 <<endl;
d --;
cout<< l1 <<endl;
d ++;
cout<< l0 <<endl;
cout<< l1 .size()<<endl;
cout<< l0 <<endl;
cout<< l1 .size()<<endl;
a .avoid( e );
d .stopAvoid();
d ++;
l0 .push_back( 0 );
d = l1 .end();
cout<< l0 <<endl;
cout<< l0 .size()<<endl;
a ++;
a .stopAvoid();
cout<< l0 <<endl;
-- e ;
e ++;
a = l1 .end();
cout<< l0 <<endl;
b --;
c = l1 .begin();
e = l1 .begin();
-- e ;
l1 .push_back( 2 );
l1 .insert( d , 1 );
e --;
a --;
a ++;
e = l0 .begin();
cout<< l1 <<endl;
-- a ;
cout<<* b <<endl;
cout<<* a <<endl;
-- b ;
a ++;
e .avoid( a );
d .avoid( e );
a ++;
b ++;
l0 .push_back( -3 );
-- a ;
c ++;
cout<< l1 <<endl;
cout<< l0 <<endl;
e .avoid( c );
cout<< l1 <<endl;
a --;
l1 .push_back( 3 );
cout<< l0 <<endl;
cout<< l0 <<endl;
cout<< l1 <<endl;
cout<< l0 .size()<<endl;
cout<< l1 <<endl;
e = l1 .begin();
cout<< l1 <<endl;
++ d ;
-- e ;
++ c ;
a .avoid( e );
```

```
cout<< l0 .size()<<endl;
cout<< l0 <<endl;
cout<< l1 <<endl;
b .avoid( d );
cout<< l0 .size()<<endl;
-- c ;
c .avoid( a );
c --;
a .avoid( b );
-- c ;
cout<<* e <<endl;
cout<<* c <<endl;
b = l1 .begin();
a ++;
cout<< l0 <<endl;
cout<< l1 <<endl;
```

Exemple de sortida 2

```
b
abcd
e
abd
1,abd
ce
1
1,abd
ce
ce
2c,e
1,abd
2c,e
1,-1,abd
1,-1,abd
1a,-1,4d,b
2
1a,-1d,4,b
1a,-1d,4,b
2c,-3,3,e
1a,-1d,4,b
3
2bc,-3,3e,
2,2bc,-3,3,e
2b,2c,-3,3,e
2,2,-3c,3,3,e
2,2,-3c,3,3,be
1,-1,4,-1,4a,-2,d
2,2,-3,3,3be,c
1,-1,4,-1,4a,-2d,
2,2,-3,3,3be,c
6
2,2,-3,3,3be,c
6
2,2,-3,3,3be,0,c
6
2,2,-3,3,3be,0,c
2,2,-3,3e,3b,0,c
1c,-1,4,-1,4,-2,2,1,ad
3
1
1,-1c,4,-1,4,-2,2,1a,d
2e,2,-3,3b,3,0,-3,
1,-1c,4,-1,4,-2,2,1a,d
2e,2,-3,3b,3,0,-3,
2e,2,-3,3b,3,0,-3,
1,-1c,4,-1,4,-2,2a,1,3,d
7
1,-1c,4,-1,4,-2,2a,1,3,d
1e,-1c,4,-1,4,-2,2a,1,3,d
7
2,2,-3,3b,3,0,-3,
1e,-1,4c,-1,4,-2,2a,1,3,d
7
1
1
2,2,-3,3,3,0,-3,
1bce,-1,4,-1,4,-2,2,1a,3,d
```

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

Autoria: PRO2

Generació: 2026-01-27T18:51:37.853Z