Push i pop d'elements marcats a la classe Stack — X16671_ca

En aquest exercici estendrem la classe `Stack` afegint un nou mètode anomenat `push_mark`. A priori, aquest mètode fa el mateix que el mètode `push` existent. També afegirem un altre nou mètode anomenat `pop_mark`. L'efecte de `pop_mark` és anar eliminant elements del cim de la pila fins que, o bé hi ha un element al cim que va ser afegit amb `push_mark`, o bé s'ha buidat la pila.
Fixeu-vos en el següent exemple de programa i en la seva execució descrita en els seus comentaris, a on denotem entre corxets els elements afegits amb `push_mark`:

```
Stack<string> s;  // s:
s.push("a");      // s: a
s.push("b");      // s: a,b
s.push_mark("c"); // s: a,b,[c]
s.push("d");      // s: a,b,[c],d
s.push("e");      // s: a,b,[c],d,e
s.push_mark("f"); // s: a,b,[c],d,e,[f]
s.push("g");      // s: a,b,[c],d,e,[f],g
s.push("h");      // s: a,b,[c],d,e,[f],g,h
s.push("i");      // s: a,b,[c],d,e,[f],g,h,i
s.pop();          // s: a,b,[c],d,e,[f],g,h
s.pop_mark();     // s: a,b,[c],d,e,[f]
s.pop_mark();     // s: a,b,[c],d,e,[f]
s.pop();          // s: a,b,[c],d,e
s.pop();          // s: a,b,[c],d
s.pop_mark();     // s: a,b,[c]
s.pop_mark();     // s: a,b,[c]
s.pop();          // s: a,b
s.pop_mark();     // s:
```

D'entre els fitxers que s'adjunten en aquest exercici, trobareu `stack.hh`, a on hi ha una implementació de la classe genèrica `Stack`. Haureu de buscar dins `stack.hh` les següents línies:

```
struct Item {
  T value;
  Item* next;
  // Afegeix el que calgui per a poder recordar si un element està marcat.
  // ...
};

...

// Modifica aquesta funció per a recordar que l'element afegit no està marcat
void push(T value) {
  Item *pnewitem = new Item();
  pnewitem->value = value;
```

```
    pnewitem->next = ptopitem;
    ptopitem = pnewitem;
    _size++;
  }

  ...

  // Pre:
  // Post: S'afegeix value al cim de la pila, com a element marcat.
  // Descomenteu les següents dues linies i implementeu el mètode:
  // void push_mark(T value) {
  // ...
  // }

  // Pre:
  // Post: S'han eliminat del cim de la pila el mínim nombre d'elements necessa
  //       per tal de garantir que o bé el cim de la pila té un element que va
  //       afegit amb push_mark, o bé la pila és buida.
  //       En particular, si el cim de la pila ja tenia un element marcat, no s
  // Descomenteu les següents dues linies i implementeu el mètode:
  // void pop_mark() {
  // ...
  // }
```

Afegiu el que cal dins de l'struct Item, i descomenteu les linies que s'indiquen i implementeu els mètodes.
D'entre els fitxers que s'adjunten a l'exercici també hi ha `main.cc` (programa principal), i el podeu compilar directament, doncs inclou `stack.hh`. Només cal que pugeu `stack.hh` al jutge.
Observació: En aquest exercici es prefereix una solució basada en manegar punters abans que una solució basada en cridar a mètodes primitius de la pròpia classe.
Observació: En els jocs de proves no es copiaran piles. Per tant, no cal que adapteu les funcions per a copiar piles, i per tant no cal decidir que passa amb els elements marcats d'una pila quan es copien en una altra pila.

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 pila `s` 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 pila:

```
push x (x és de tipus T)
pop
top
size
print
push_mark x (x és de tipus T)
pop_mark
```

Se suposa que la seqüència d'entrada serà correcta (sense pop ni top sobre pila buida).
El programa principal que us oferim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe pila. Només cal que feu els canvis abans esmentats.

Sortida

Per a cada instrucció `top`, s'escriurà el top actual de la pila, per a cada instrucció `print`, s'escriurà el contingut de la pila, i per a cada instrucció `size`, s'escriurà la mida de la pila. El programa que us oferim ja fa això. Només cal que implementeu els mètodes abans esmentats i les modificacions que considereu oportunes a la classe.

Exemple d'entrada 1

```
pop_mark
size
pop_mark
push_mark z
pop_mark
size
pop
pop_mark
size
push a
push b
size
pop_mark
size
pop_mark
size
push_mark a
top
print
size
push b
top
print
size
push c
top
print
size
push d
top
print
size
push_mark bb
top
print
size
push ccc
top
print
size
push dd
top
print
size
push ee
top
print
size
pop
top
print
size
pop_mark
top
print
size
pop_mark
top
print
size
pop_mark
top
print
size
pop
top
print
size
pop_mark
top
print
size
push dd
top
print
size
push eeee
top
print
size
pop_mark
top
print
size
pop
print
size
```

Exemple de sortida 1

```
0
1
0
2
0
0
a
a
1
b
a b
2
c
a b c
3
d
a b c d
4
bb
a b c d bb
5
ccc
a b c d bb ccc
6
dd
a b c d bb ccc dd
7
ee
a b c d bb ccc dd ee
8
dd
a b c d bb ccc dd
7
bb
a b c d bb
5
bb
a b c d bb
5
d
a b c d
4
a
a
1
dd
a dd
2
eeee
a dd eeee
3
a
a
1

0
```

Exemple d'entrada 2

```
push b
push b
push_mark ca
top
size
top
push_mark ad
size
push c
pop
pop
push aab
push ccb
push bb
top
push a
pop_mark
top
push bbc
size
size
pop
push c
push cb
push bd
pop
size
push_mark ddb
push_mark c
pop_mark
push db
push c
size
top
top
pop
push ca
size
push_mark b
top
size
size
size
push bc
push dcc
push ac
top
top
push cc
push aa
pop
size
size
size
pop
push db
size
top
push da
```

```
pop
size
top
push c
size
push a
push dd
top
push ddb
push acd
push_mark d
push_mark c
push a
pop_mark
top
push a
pop
push cbc
size
top
pop_mark
print
```

Exemple de sortida 2

```
ca
3
ca
4
bb
ca
4
4
5
9
c
c
9
b
10
10
10
ac
ac
14
14
14
14
db
14
db
15
dd
c
22
cbc
b b ca c cb ddb c db ca b bc dcc ac db c a dd ddb acd c
```

Observació

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

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

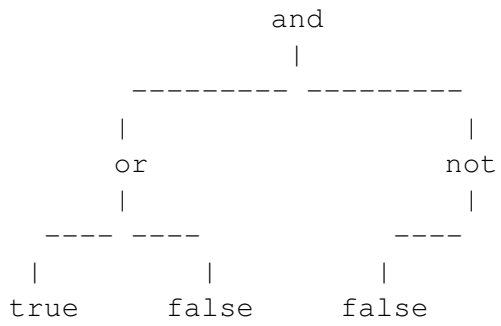
Entenem com a solució ràpida una que és correcta i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics. Per exemple, una solució que superi tots els jocs de proves però que manegui incorrectament la memòria serà invalidada i tindrà nota 0.
Una solució basada en cridar a mètodes primitius de la pròpia classe pot tenir una certa penalització en la nota.

Informació del problema

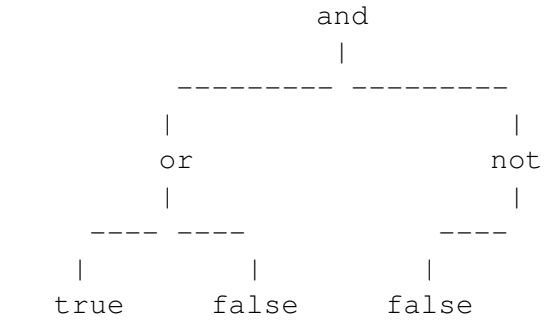
Autor : PRO2
Generació : 2024-01-11 11:52:27

Avaluar expressions booleanes X45696_ca

INTRODUCCIÓ:
En aquest exercici considerarem arbres d'strings que representen expressions booleanes sobre valors **true, false** i els operadors booleans **and, or, not**. En el cas de **not**, que és un operador amb un sol operand, considerarem que aquest operand és sempre el fill esquerre. Per exemple, el següent arbre representa l'expressió **(true or false) and (not(false))**. Aquesta expressió s'avalua a **true**.



EXERCICI:
Afegiu un mètode a la classe Arbre que, quan el paràmetre implícit sigui un arbre binari d'strings que representa una expressió booleana correcta sobre **true,false** i operadors **and,or,not**, retorni la seva avaluació.
Mostrem un exemple de paràmetre d'entrada de la funció i la corresponent sortida. Si l'arbre *a* és



llavors tenim que la crida `a.evaluate()` retornarà `true`.
D'entre els fitxers que s'adjunten en aquest exercici, trobareu `Arbre.hh`, a on hi ha una implementació de la classe genèrica Arbre binari. Haureu de buscar dins `Arbre.hh` les següents línies:

```
// Pre:  el, p.i. és un arbre no buit que representa una expressió booleana cor
//       sobre els valors true,false i els operadors and,or,not.
// Post: Retorna l'avaluació de l'expressió representada pel p.i.
// Descomenteu les següents dues línies i implementeu el mètode:
//  bool evaluate(){
// }
```

Descomenteu les dues línies que s'indiquen i implementeu el mètode, fent servir l'operació privada que trobareu just després i que també haureu d'implementar. No toqueu la resta de la implementació de la classe.
La implementació d'aquest mètode hauria de consistir en accedir a nodes mitçançant punters. De fet, possiblement qualsevol altra implementació produirà error d'execució.
Observació: Per a superar els jocs de proves privats convindrà tenir en compte aquestes optimitzacions típiques d'expressions booleanes:

- Si una expressió e_1 s'avalua a `false`, llavors l'avaluació de (e_1 `and` e_2) és `false` i no requereix avaluar e_2.
- Si una expressió e_1 s'avalua a `true`, llavors l'avaluació de (e_1 `or` e_2) és `true` i no requereix avaluar e_2.

De fet, les operacions `and` i `or` de `C++` ja optimitzen així, de manera que una solució senzilla i natural hauria de poder 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 `Arbre.hh`. Només cal que pugeu `Arbre.hh` al jutge.

Entrada

Un nombre arbitrari d'arbres. Cada cas consisteix en una descripció d'un arbre binari que representa una expressió booleana correcta. La descripció consisteix en un recorregut en preordre del nodes de l'arbre, amb marques on hi anirien els arbres buits. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes entrades. Només cal que implementeu la funció abans esmentada.

Sortida

Per a cada cas, la sortida conté la corresponent avaluació de l'arbre. Fixeu-vos en que el programa que us oferim ja s'encarrega d'escriure aquesta avaluació. Només cal que implementeu la funció abans esmentada.

Exemple d'entrada 1

```
and and and false # # true # # or false # # false # # and true # # false # #
or and true # # true # # not true # # #
and not true # # # and false # # false # #
false # #
or or and true # # true # # false # # or true # # true # #
not or or true # # false # # true # # #
or false # # or not false # # # and false # # false # #
and or or and false # # true # # and true # # true # # and true # # true # # and o
or true # # false # #
or or and true # # false # # or and false # # true # # or true # # false # # true # #
#
```

Exemple de sortida 1

```
false
true
false
true
false
true
true
true
```

Exemple d'entrada 2

```
and and and false # # true # # or false # # false # # and true # # false # #
or and true # # true # # not true # # #
and not true # # # and false # # false # #
false # #
or or and true # # true # # false # # or true # # true # #
not or or true # # false # # true # # #
or false # # or not false # # # and false # # false # #
and or or and false # # true # # and true # # true # #
or true # # false # #
or or and true # # false # # or and false # # true # #
#
```

Exemple de sortida 2

```
false
true
false
true
false
true
true
true
```

Exemple de sortida 2

```
false
true
false
false
```

```
true
false
true
true
true
true
```

Observació

La vostra funció i subfuncions que creeu han de treballar només amb arbres, punters i nodes d'arbre. Heu de trobar una solució **RECURSIVA** del problema.
Avaluació sobre 10 punts:

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

Entenem com a solució ràpida una que és correcta, de cost lineal i que optimitza operacions booleanes, 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 : 2024-01-10 15:57:54

Suma i mida de molts arbres X68237_ca

En aquest exercici, heu d'implementar un programa que llegeix comandes que manipulen variables que guarden àrbres binaris d'enters. La primera comanda `numvars=` n `;` indica el nombre total n de variables. Els noms d'aquestes variables son `t0,...,t(n-1)`, i se suposa que inicialment cadascuna guarda un àrbre buit. Després venen comandes que construeixen nous àrbres a partir de variables i els assignen a variables (com per exemple `t2 =BinTree( 3 , t0 , t1 );`, i comandes que accedeixen als fills d'un arbre existent i els assignen a variables (com per exemple `t3 = t2 .left();` o `t3 = t2 .right();`). També hi ha comandes per a escriure per la sortida un àrbre en `INLINEFORMAT` (com per exemple `cout<< t2 ;`), i instruccions per a escriure la mida o la suma dels valors d'un arbre guardat en una variable, com per exemple (`cout<<size( t2 )<<endl;` o `cout<<sum( t2 );`).
Aquest és un exemple d'entrada del programa:

```
numvars= 4 ;
t1 =BinTree( 1 , t2 , t3 );
t2 =BinTree( 2 , t1 , t3 );
t3 =BinTree( 3 , t2 , t1 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<size( t0 )<<endl;
cout<<size( t1 )<<endl;
cout<<size( t2 )<<endl;
cout<<size( t3 )<<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t1 )<<endl;
cout<<sum( t2 )<<endl;
cout<<sum( t3 )<<endl;
t1 =BinTree( 1 , t2 , t3 );
t2 =BinTree( 2 , t1 , t3 );
t3 =BinTree( 3 , t2 , t1 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<size( t0 )<<endl;
cout<<size( t1 )<<endl;
cout<<size( t2 )<<endl;
cout<<size( t3 )<<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t1 )<<endl;
cout<<sum( t2 )<<endl;
cout<<sum( t3 )<<endl;
t1 = t3 .left();
```

```
t2 = t1 .right();
t3 = t2 .left();
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<size( t0 )<<endl;
cout<<size( t1 )<<endl;
cout<<size( t2 )<<endl;
cout<<size( t3 )<<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t1 )<<endl;
cout<<sum( t2 )<<endl;
cout<<sum( t3 )<<endl;
```

La sortida del programa amb la seqüència de comandes d'entrada anterior hauria de ser:

```
()
1
2(1,)
3(2(1,),1)
0
1
2
4
0
1
3
7
()
1(2(1,),3(2(1,),1))
2(1(2(1,),3(2(1,),1)),3(2(1,),1))
3(2(1(2(1,),3(2(1,),1)),3(2(1,),1)),1(2(1,),3(2(1,),1)))
0
7
12
20
0
11
20
34
()
2(1(2(1,),3(2(1,),1)),3(2(1,),1))
3(2(1,),1)
2(1,)
0
12
4
2
0
```

```
20
7
3
```

Com podeu observar a l'exemple d'entrada anterior, hi han espais en blanc per a facilitar la lectura. Podeu llegir i tractar les comandes així:

```
#include <iostream>
#include <string>
#include <cstdlib>
//...

using namespace std;

#include "BinTree.hh"

int getIdVar(string s)
{
return atoi(s.substr(1).c_str());
}

//...

int main()
{
//...
string s1, s2, s3, s4, s5, s6, s7;
int numvars;
cin >> s1 >> numvars >> s2;
// ...
while (cin >> s1 >> s2) {
if (s1[0] == 't') {
int idvar = getIdVar(s1);
if (s2 == "=BinTree(") {
int value;
cin >> value >> s3 >> s4 >> s5 >> s6 >> s7;
int idvar1 = getIdVar(s4);
int idvar2 = getIdVar(s6);
//...
} else if (s2 == "=") {
cin >> s3 >> s4;
int idvar1 = getIdVar(s3);
if (s4 == ".left();") {
//...
} else {
//...
}
}
} else if (s1 == "cout<<") {
int idvar = getIdVar(s2);
```

```
cin >> s3;
//...
//....setOutputFormat(BinTree<int>::INLINEFORMAT);
//cout << ... << endl;
} else if (s1 == "cout<<size(") {
int idvar = getIdVar(s2);
cin >> s3;
//...
} else if (s1 == "cout<<sum(") {
int idvar = getIdVar(s2);
cin >> s3;
//...
}
}
}
```

Fixeu-vos que l'enunciat d'aquest exercici us ofereix el fitxer `BinTree.hh`. Us falta crear el fitxer `main.cc`, que haurieu de construïr a partir de la plantilla que us hem oferit abans, fent un ús convenient del tipus `BinTree`. Només cal que pugeu `main.cc` al jutge.
Observació: Us recomanem que comenceu implementant una solució bàsica per tal de superar els jocs de proves públics i obtenir així la meitat de la nota. Ja la optimitzareu més endavant si teniu temps.

Entrada

La primera linia de l'entrada és de la forma `numvars= LIMIT ;`, a on LIMIT és un nombre natural positiu. Després venen instruccions d'aquestes menes:

```
tNUM =BinTree( VALUE , tNUM1 , tNUM2 );
tNUM1 = tNUM2 .left();
tNUM1 = tNUM2 .right();
cout<< tNUM <<endl;
cout<<size( tNUM )<<endl;
cout<<sum( tNUM )<<endl;
```

On `VALUE` es un enter i `NUM, NUM1, NUM2` son naturals en el rang $\{0,\ldots,\text{LIMIT-1}\}$.
Se suposa que les entrades son correctes: sempre es demana accedir a `left` o `right` d'arbres no buits, i no es produeixen errors d'overflow.

Sortida

Per a cada instrucció dels següents tres tipus, el vostre programa ha d'escriure el resultat esperat (l'arbre contingut en la variable en `INLINEFORMAT`, o la mida de l'arbre contingut en la variable, o la suma de l'arbre contingut en la variable, segons el cas).

```
cout<< tNUM <<endl;
cout<<size( tNUM )<<endl;
cout<<sum( tNUM )<<endl;
```

Exemple d'entrada 1

```
numvars= 4 ;
t1 =BinTree( 1 , t2 , t3 );
t2 =BinTree( 2 , t1 , t3 );
t3 =BinTree( 3 , t2 , t1 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<size( t0 )<<endl;
cout<<size( t1 )<<endl;
cout<<size( t2 )<<endl;
cout<<size( t3 )<<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t1 )<<endl;
cout<<sum( t2 )<<endl;
cout<<sum( t3 )<<endl;
t1 =BinTree( 1 , t2 , t3 );
t2 =BinTree( 2 , t1 , t3 );
t3 =BinTree( 3 , t2 , t1 );
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<size( t0 )<<endl;
cout<<size( t1 )<<endl;
cout<<size( t2 )<<endl;
cout<<size( t3 )<<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t1 )<<endl;
cout<<sum( t2 )<<endl;
cout<<sum( t3 )<<endl;
t1 = t3 .left();
t2 = t1 .right();
t3 = t2 .left();
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<<size( t0 )<<endl;
cout<<size( t1 )<<endl;
cout<<size( t2 )<<endl;
cout<<size( t3 )<<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t1 )<<endl;
cout<<sum( t2 )<<endl;
cout<<sum( t3 )<<endl;
```

Exemple de sortida 1

```
()
1
2(1,)
3(2(1,),1)
0
1
2
4
0
1
3
7
()
1(2(1,),3(2(1,),1))
2(1(2(1,),3(2(1,),1)),3(2(1,),1))
3(2(1(2(1,),3(2(1,),1)),3(2(1,),1)),1(2(1,),3(2(1,),1)
0
7
12
20
0
11
20
34
()
2(1(2(1,),3(2(1,),1)),3(2(1,),1))
3(2(1,),1)
2(1,)
0
12
4
2
0
20
7
3
```

Exemple d'entrada 2

```
numvars= 3 ;
cout<< t1 <<endl;
cout<< t1 <<endl;
t1 =BinTree( 1 , t0 , t0 );
t1 =BinTree( 2 , t2 , t1 );
cout<<size( t1 )<<endl;
t0 = t1 .left();
t2 =BinTree( 5 , t0 , t0 );
t2 =BinTree( 2 , t1 , t1 );
t0 = t2 .right();
t1 = t1 .left();
t0 =BinTree( 4 , t0 , t0 );
t1 =BinTree( 2 , t2 , t2 );
cout<<size( t0 )<<endl;
cout<<sum( t1 )<<endl;
t2 =BinTree( 2 , t0 , t2 );
cout<< t1 <<endl;
cout<<size( t0 )<<endl;
t0 = t1 .right();
t0 =BinTree( 0 , t0 , t0 );
t1 = t0 .left();
t2 = t1 .right();
```

```
cout<< t0 <<endl;
t0 = t1 .left();
t0 =BinTree( 4 , t0 , t2 );
cout<<sum( t1 )<<endl;
cout<<sum( t1 )<<endl;
cout<< t0 <<endl;
t0 =BinTree( 2 , t0 , t1 );
t2 =BinTree( 1 , t2 , t1 );
cout<< t2 <<endl;
cout<<sum( t2 )<<endl;
t1 = t2 .right();
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
t2 = t0 .right();
cout<< t1 <<endl;
cout<<size( t1 )<<endl;
cout<<sum( t0 )<<endl;
cout<<size( t2 )<<endl;
t1 = t2 .right();
cout<<sum( t1 )<<endl;
cout<<size( t1 )<<endl;
t2 = t1 .left();
cout<< t1 <<endl;
t1 =BinTree( 3 , t1 , t2 );
cout<<sum( t2 )<<endl;
t1 =BinTree( 2 , t1 , t1 );
cout<< t2 <<endl;
cout<<sum( t0 )<<endl;
cout<<size( t2 )<<endl;
t1 =BinTree( 5 , t2 , t1 );
cout<< t2 <<endl;
t2 = t1 .right();
cout<<sum( t0 )<<endl;
t2 = t1 .left();
t2 =BinTree( 1 , t2 , t1 );
cout<< t2 <<endl;
cout<< t2 <<endl;
cout<<size( t1 )<<endl;
cout<<sum( t1 )<<endl;
cout<< t1 <<endl;
t1 = t2 .left();
t1 = t0 .right();
cout<<sum( t1 )<<endl;
t2 = t1 .right();
cout<< t1 <<endl;
t1 =BinTree( 2 , t1 , t2 );
cout<< t2 <<endl;
t2 = t0 .right();
t2 = t0 .right();
cout<<size( t1 )<<endl;
cout<< t1 <<endl;
cout<< t1 <<endl;
cout<< t1 <<endl;
cout<<sum( t1 )<<endl;
t1 = t1 .left();
cout<<size( t2 )<<endl;
cout<< t1 <<endl;
cout<< t0 <<endl;
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
```

Exemple de sortida 2

```
()
()
2
5
18
2(2(2(,1),2(,1)),2(2(,1),2(,1)))
5
0(2(2(,1),2(,1)),2(2(,1),2(,1)))
8
8
4(2(,1),2(,1))
1(2(,1),2(2(,1),2(,1)))
12
2(4(2(,1),2(,1)),2(2(,1),2(,1)))
2(2(,1),2(,1))
1(2(,1),2(2(,1),2(,1)))
2(2(,1),2(,1))
5
20
5
3
2
2(,1)
0
()
20
0
()
20
1(,5(,2(3(2(,1),),3(2(,1),))))
1(,5(,2(3(2(,1),),3(2(,1),))))
8
19
5(,2(3(2(,1),),3(2(,1),)))
8
2(2(,1),2(,1))
2(,1)
8
2(2(2(,1),2(,1)),2(,1))
2(2(2(,1),2(,1)),2(,1))
2(2(2(,1),2(,1)),2(,1))
13
5
2(2(,1),2(,1))
2(4(2(,1),2(,1)),2(2(,1),2(,1)))
2(4(2(,1),2(,1)),2(2(,1),2(,1)))
2(2(,1),2(,1))
2(2(,1),2(,1))
```

Exemple d'entrada 3

```
numvars= 10 ;
cout<< t6 <<endl;
cout<< t5 <<endl;
t6 =BinTree( -1 , t2 , t9 );
t7 =BinTree( 7 , t0 , t9 );
cout<<size( t6 )<<endl;
t8 =BinTree( 6 , t7 , t9 );
t2 =BinTree( -15 , t3 , t7 );
t9 = t2 .right();
t8 = t9 .right();
t3 = t6 .left();
t9 =BinTree( -1 , t3 , t1 );
t7 =BinTree( 4 , t8 , t4 );
cout<<size( t0 )<<endl;
cout<<sum( t6 )<<endl;
t6 =BinTree( 13 , t3 , t2 );
cout<< t6 <<endl;
cout<<size( t5 )<<endl;
t7 =BinTree( 6 , t6 , t5 );
t5 =BinTree( 9 , t2 , t5 );
t4 =BinTree( -2 , t4 , t3 );
t8 =BinTree( 6 , t4 , t3 );
t4 = t9 .right();
cout<< t0 <<endl;
t8 = t9 .left();
t6 =BinTree( -10 , t4 , t9 );
cout<<sum( t0 )<<endl;
cout<<sum( t8 )<<endl;
cout<< t1 <<endl;
t7 =BinTree( 18 , t2 , t2 );
t0 =BinTree( -7 , t6 , t1 );
cout<< t9 <<endl;
cout<<sum( t9 )<<endl;
t1 =BinTree( 20 , t7 , t7 );
t5 =BinTree( 0 , t9 , t7 );
t6 = t7 .right();
cout<< t6 <<endl;
cout<< t6 <<endl;
t4 =BinTree( 6 , t8 , t1 );
cout<< t9 <<endl;
t9 = t0 .right();
cout<< t8 <<endl;
cout<<size( t0 )<<endl;
cout<<sum( t6 )<<endl;
t5 =BinTree( 18 , t6 , t1 );
cout<<size( t5 )<<endl;
t8 = t4 .right();
cout<<sum( t1 )<<endl;
cout<<size( t3 )<<endl;
t4 = t4 .left();
cout<< t4 <<endl;
t3 =BinTree( 5 , t1 , t7 );
t6 =BinTree( 8 , t2 , t1 );
t5 =BinTree( -11 , t7 , t4 );
cout<<sum( t8 )<<endl;
t7 =BinTree( 19 , t5 , t3 );
t3 =BinTree( 12 , t1 , t8 );
t4 =BinTree( 19 , t3 , t3 );
cout<< t8 <<endl;
t4 =BinTree( -9 , t8 , t8 );
t7 =BinTree( 2 , t7 , t6 );
cout<<sum( t3 )<<endl;
cout<<size( t3 )<<endl;
t2 =BinTree( -9 , t5 , t4 );
cout<< t5 <<endl;
t6 =BinTree( -20 , t9 , t2 );
t4 = t7 .right();
t4 =BinTree( -6 , t8 , t1 );
t9 =BinTree( 8 , t3 , t6 );
t2 =BinTree( -18 , t1 , t0 );
t1 =BinTree( 9 , t0 , t8 );
t6 =BinTree( 15 , t4 , t6 );
t8 =BinTree( -13 , t6 , t2 );
t7 =BinTree( 7 , t2 , t4 );
cout<<sum( t6 )<<endl;
t9 =BinTree( 18 , t0 , t8 );
t1 =BinTree( -4 , t1 , t0 );
t4 = t0 .left();
t1 =BinTree( -12 , t9 , t6 );
t3 =BinTree( -15 , t8 , t0 );
cout<< t6 <<endl;
cout<< t4 <<endl;
t4 =BinTree( 0 , t6 , t2 );
cout<<size( t7 )<<endl;
t9 =BinTree( -7 , t8 , t7 );
t2 =BinTree( -2 , t9 , t9 );
t2 =BinTree( 9 , t7 , t6 );
cout<<sum( t3 )<<endl;
cout<< t1 <<endl;
t9 =BinTree( -6 , t1 , t4 );
t1 = t0 .left();
t8 =BinTree( -7 , t7 , t0 );
t8 = t0 .right();
cout<<sum( t2 )<<endl;
t6 = t1 .right();
t2 =BinTree( -4 , t2 , t2 );
cout<< t5 <<endl;
t9 =BinTree( 9 , t0 , t2 );
cout<< t3 <<endl;
t0 = t4 .right();
t9 = t1 .right();
cout<<size( t6 )<<endl;
cout<< t5 <<endl;
cout<< t4 <<endl;
t3 =BinTree( -18 , t6 , t0 );
t2 =BinTree( 1 , t9 , t4 );
cout<< t5 <<endl;
cout<<sum( t7 )<<endl;
t1 =BinTree( -10 , t4 , t6 );
t4 = t2 .left();
cout<<size( t6 )<<endl;
cout<< t1 <<endl;
cout<< t8 <<endl;
cout<< t9 <<endl;
t8 = t3 .right();
t3 = t8 .right();
t7 =BinTree( 6 , t6 , t8 );
t6 =BinTree( 20 , t8 , t7 );
t3 =BinTree( 9 , t3 , t3 );
t2 =BinTree( 18 , t6 , t5 );
```

```
cout<<sum( t6 )<<endl;
t7 =BinTree( 16 , t9 , t6 );
t1 =BinTree( 9 , t0 , t4 );
t0 =BinTree( -19 , t8 , t6 );
t4 = t7 .left();
t9 =BinTree( 20 , t2 , t8 );
t0 = t1 .left();
t8 = t9 .right();
cout<< t5 <<endl;
t2 =BinTree( 11 , t5 , t7 );
t9 =BinTree( -7 , t4 , t5 );
cout<< t3 <<endl;
t7 = t0 .right();
t1 = t9 .left();
cout<<sum( t7 )<<endl;
cout<<size( t2 )<<endl;
cout<< t3 <<endl;
t9 = t0 .right();
t1 =BinTree( 13 , t9 , t6 );
cout<< t1 <<endl;
cout<<sum( t0 )<<endl;
cout<<sum( t3 )<<endl;
t1 = t9 .left();
t6 =BinTree( 14 , t7 , t7 );
cout<<size( t0 )<<endl;
t7 =BinTree( 6 , t4 , t8 );
t2 =BinTree( -14 , t6 , t0 );
t6 = t4 .right();
t8 =BinTree( 7 , t9 , t6 );
cout<< t7 <<endl;
t0 =BinTree( -15 , t1 , t3 );
t7 = t2 .right();
cout<< t4 <<endl;
t0 =BinTree( 16 , t2 , t9 );
cout<<sum( t9 )<<endl;
t4 = t5 .right();
t3 =BinTree( 19 , t7 , t8 );
cout<<size( t6 )<<endl;
t6 =BinTree( -7 , t7 , t6 );
t7 =BinTree( 0 , t5 , t9 );
cout<<size( t8 )<<endl;
t3 =BinTree( -7 , t8 , t3 );
t3 =BinTree( -7 , t7 , t8 );
t1 =BinTree( 1 , t9 , t0 );
cout<< t8 <<endl;
cout<<sum( t5 )<<endl;
t4 = t3 .right();
cout<< t1 <<endl;
t0 =BinTree( -18 , t9 , t7 );
cout<<size( t9 )<<endl;
cout<< t2 <<endl;
cout<<sum( t3 )<<endl;
t2 =BinTree( -2 , t2 , t0 );
cout<<size( t1 )<<endl;
cout<< t5 <<endl;
t4 =BinTree( 9 , t1 , t7 );
t1 = t1 .right();
t4 =BinTree( -3 , t0 , t8 );
t1 = t6 .left();
t1 =BinTree( 3 , t9 , t6 );
t8 = t0 .left();
t1 = t0 .left();
t9 =BinTree( 12 , t7 , t5 );
t4 = t6 .right();
t9 =BinTree( -9 , t7 , t4 );
cout<<size( t1 )<<endl;
cout<< t0 <<endl;
t8 =BinTree( 19 , t5 , t3 );
cout<<sum( t4 )<<endl;
t5 =BinTree( -12 , t4 , t9 );
cout<<sum( t3 )<<endl;
t2 =BinTree( -16 , t2 , t2 );
t3 =BinTree( 13 , t4 , t6 );
t3 =BinTree( 20 , t3 , t4 );
t3 = t9 .right();
cout<< t2 <<endl;
t8 = t9 .left();
t4 =BinTree( -10 , t7 , t6 );
cout<<size( t7 )<<endl;
cout<<sum( t3 )<<endl;
t3 = t8 .left();
cout<< t0 <<endl;
cout<< t1 <<endl;
cout<< t2 <<endl;
cout<< t3 <<endl;
cout<< t4 <<endl;
cout<< t5 <<endl;
cout<< t6 <<endl;
cout<< t7 <<endl;
cout<< t8 <<endl;
cout<< t9 <<endl;
```

Exemple de sortida 3

```
()
()
1
0
-1
13(,-15(,7))
0
()
0
0
()
-1
-1
-15(,7)
-15(,7)
-1
()
3
-8
14
24
0
()
24
20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7)))
60
23
-11(18(-15(,7),-15(,7)),)
58
15(-6(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7)))),
-10(,-1)
39
0
-12(18(-7(-10(,-1),),-13(15(-6(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),20(18(-15(,7),-15(,7)),
104
-11(18(-15(,7),-15(,7)),)
-15(-13(15(-6(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7)))),
1
-11(18(-15(,7),-15(,7)),)
0(15(-6(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7)))),
-11(18(-15(,7),-15(,7)),)
37
```

```
1
-10(0(15(-6(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),
()
-1
1
-11(18(-15(,7),-15(,7)),)
9(-7(-10(,-1),),-7(-10(,-1),))
-18
42
9(-7(-10(,-1),),-7(-10(,-1),))
13(-7(-10(,-1),),20(-18(20(18(-15(,7),-15(,7)),18(-15(
-12
-27
15
6(-1,-18(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),-
-1
-18
0
4
7(-7(-10(,-1),),)
-9
1(-7(-10(,-1),),16(-14(14(-7(-10(,-1),),-7(-10(,-1),))
3
-14(14(-7(-10(,-1),),-7(-10(,-1),)),-18(20(18(-15(,7),-
-45
31
-11(18(-15(,7),-15(,7)),)
3
-18(-7(-10(,-1),),0(-11(18(-15(,7),-15(,7)),),-7(-10(,
0
-45
-16(-2(-14(14(-7(-10(,-1),),-7(-10(,-1),)),-18(20(18(-
10
0
-18(-7(-10(,-1),),0(-11(18(-15(,7),-15(,7)),),-7(-10(,
-7(-10(,-1),))
-16(-2(-14(14(-7(-10(,-1),),-7(-10(,-1),)),-18(20(18(-
-11(18(-15(,7),-15(,7)),)
-10(0(-11(18(-15(,7),-15(,7)),),-7(-10(,-1),)),-7(-18(
-12(,-9(0(-11(18(-15(,7),-15(,7)),),-7(-10(,-1),)),))
-7(-18(20(18(-15(,7),-15(,7)),18(-15(,7),-15(,7))),-7(
0(-11(18(-15(,7),-15(,7)),),-7(-10(,-1),))
-9(0(-11(18(-15(,7),-15(,7)),),-7(-10(,-1),)),)
```

Observació

La solució d'aquest exercici s'ha de basar en un ús raonable del tipus `BinTree`. Qualsevol solució que ignori això i faci servir enfocaments o estructures de dades alternatives que no formen part de l'assignatura serà invalidada.
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**, 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 : 2024-01-11 12:40:30

++ i – amb rebot a la classe List — X94330_ca

Típicament, l'operador ++ dels iteradors de la classe List els desplaça una unitat cap al end de la llista, i l'operador -- dels iteradors de la classe List els desplaça una unitat cap al begin de la llista. A més a més, 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.
En aquest exercici modificarem la classe List de manera que els errors d'execució abans esmentats ja no es produiran. En canvi, es produirà un intercanvi en la direcció de moviment dels operadors ++ i -- (efecte rebot). Per exemple, si creem un iterador nou, el col.loquem al end de la llista, i executem ++ sobre ell, no hi haurà error d'execució, l'iterador no es mourà, i a partir d'aquell moment l'operador ++ sobre ell l'anirà desplaçant cap al begin de la llista, i l'operador -- el desplaçarà cap al end de la llista.
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
List<string>::iterator it = l.begin(); // l: (a),b,c
it++;                                    // l: a,(b),c
it++;                                    // l: a,b,(c)
it--;                                    // l: a,(b),c
it++;                                    // l: a,b,(c)
it++;                                    // l: a,b,c,()
it++;                                    // l: a,b,c,()
it++;                                    // l: a,b,(c)
it++;                                    // l: a,(b),c
it--;                                    // l: a,b,(c)
it++;                                    // l: a,(b),c
it++;                                    // l: (a),b,c
it++;                                    // l: (a),b,c
it++;                                    // l: a,(b),c
it--;                                    // l: (a),b,c
it--;                                    // l: (a),b,c
it--;                                    // l: a,(b),c
it--;                                    // l: a,b,(c)
it++;                                    // l: a,(b),c
it--;                                    // l: a,b,(c)
it--;                                    // l: a,b,c,()
it--;                                    // l: a,b,c,()
it++;                                    // l: a,b,c,()
it++;                                    // l: a,b,(c)
```

D'entre els fitxers que s'adjunten en aquest exercici, trobareu `list.hh`, a on hi ha una implementació de la classe genèrica `List`. Caldrà que modifiqueu la classe `List` per a poder

recordar quina és la direcció actual de desplaçament d'un iterador donat, i reimplementeu els operadors ++ i -- convenientment. Més específicament, haureu de buscar les següents línies i fer els afegits i modificacions que s'hi indiquen:

```
  ...

  // Iterators mutables
  class iterator {
    friend class List;
    private:
      List *plist;
      Item *pitem;
      // Add an attribute to remember the orientation of the iterator:
      // ...

      // You can add new private methods if you wish.

    public:

    iterator() {
        // Initialise the orientation of the iterator
        // ...
    }

    // Adapt this function so that no error occurs and the orientation of the i
    // is taken into account and updated 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: ++ on iterator at the end of list" << endl;
        exit(1);
      }
      pitem = pitem->next;
      return *this;
    }

...

    // Adapt this function so that no error occurs and the orientation of the i
    // is taken into account and updated 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: -- on iterator at the beginning of list" << endl;
       exit(1);
      }
      pitem = pitem->prev;
      return *this;
    }

...
```

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
it2 = begin
it2 = end
it2 = erase it2
it2++
it2--
++it2
--it2
*it2 = x (x és string)
insert it2 x (x és string)
cout << *it2
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 les modificacions abans esmentades.

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++
cout << l
push_back a
push_back b
push_back c
cout << l
it1 = begin
cout << l
it1--
cout << l
--it1
cout << l
it1--
cout << l
++it1
cout << l
--it1
cout << l
it1--
cout << l
it1++
cout << l
it1--
cout << l
it1--
cout << l
it1--
cout << l
it1++
cout << l
it1++
cout << l
it1++
cout << l
it1--
cout << l
it1 = begin
it2 = end
cout << l
it2++
cout << l
it2++
cout << l
cout << *it1
cout << *it2
push_front i
push_back j
push_front k
push_back l
cout << l
it1 = end
it1++
it1--
--it1
it1--
it2++
++it2
it2--
--it2
--it2
cout << l
it1 = begin
it2 = end
it2++
++it2
cout << l
cout << *it1
cout << *it2
it2++
++it2
it1--
it1--
cout << l
cout << *it1
cout << *it2
insert it1 o
insert it2 p
cout << l
pop_front
pop_back
cout << l
it1--
--it1
it2++
cout << l
it1 = erase it1
cout << l
```

```
++it2
cout << l
it2 = erase it2
cout << l
*it1 = x
cout << l
*it2 = y
cout << l
it1++
++it2
cout << l
it1--
it2--
cout << l
```

Exemple de sortida 1

```
([])
([])
a b c ([])
(a) b c []
(a) b c []
a (b) c []
a b (c) []
a (b) c []
a b (c) []
a b c ([])
a b (c) []
a b c ([])
a b c ([])
a b (c) []
a b c ([])
a b c ([])
a b (c) []
a b c ([])
(a) b c []
(a) b c []
(a) b [c]
a
c
k i (a) b [c] j l
k i a b c ([j]) l
(k) i a b c j [l]
k
l
k (i) a b [c] j l
i
c
k o (i) a b p [c] j l
o (i) a b p [c] j
o i a (b) [p] c j
o i a ([p]) c j
o i [a] (p) c j
o i ([p]) c j
o i ([x]) c j
o i ([y]) c j
o ([i]) y c j
o i ([y]) c j
```

Exemple d'entrada 2

```
push_front bi
it2--
it2 = begin
cout << *it2
++it1
it2++
it2--
cout << *it2
push_back yqs
it2++
it2 = end
insert it2 gfn
--it1
it2 = begin
++it2
```

```
pop_back
push_front lf
push_back t
it2 = end
push_front dec
push_front tj
it1--
++it1
push_back p
push_front am
push_front rpqe
it2++
push_front ssg
--it2
push_back e
push_front y
push_back pgjh
```

```
insert it1 flh
it2--
push_back lmfd
--it2
insert it1 e
push_front p
insert it1 rq
cout << *it2
--it1
cout << l
cout << *it1
push_front tm
cout << *it2
cout << *it1
pop_front
*it2 = wrl
it1++
it1--
insert it2 gfou
++it1
++it1
it1 = end
insert it2 wp
push_front a
push_front g
it2++
it2++
push_front g
it1--
++it2
++it1
push_back dlx
push_front rrg
it2++
++it1
it2++
push_front a
insert it2 jnec
insert it2 ac
++it2
++it2
push_front d
cout << l
it1--
push_front zq
insert it1 xy
--it2
--it2
cout << l
it1--
++it1
it2--
push_front gln
insert it1 k
--it1
cout << *it2
it2++
it1++
++it1
--it2
++it1
```

```
it1++
cout << *it2
pop_back
it2 = end
cout << *it1
--it1
--it1
push_back pkfc
push_front qivz
push_front yhj
it2--
push_front av
--it1
cout << *it1
++it1
cout << *it2
it1++
it2 = begin
++it2
push_back ywb
pop_back
--it1
it1++
insert it2 nma
it2 = erase it2
++it2
push_back vvv
--it2
cout << *it2
++it1
push_front j
insert it1 cn
push_back rca
*it2 = s
push_back ouv
push_front mgm
insert it1 trm
cout << *it1
it2 = begin
++it1
it2--
*it1 = a
insert it1 ya
it1 = begin
cout << *it2
push_front fw
insert it1 xzm
push_back ibxw
++it2
it2 = erase it2
insert it2 mkru
it1++
push_front vb
++it1
--it1
--it2
it2--
--it1
cout << *it1
it2--
insert it1 e
```

```
cout << *it1
it1--
push_front iksl
insert it2 bgsb
cout << *it2
push_front s
cout << *it1
++it1
push_back ios
push_back pkls
*it2 = j
it1++
cout << *it1
insert it1 mzrn
++it2
cout << *it1
push_back lyfe
cout << *it2
cout << *it2
push_front fg
--it1
cout << *it1
it1--
cout << *it1
push_back l
push_front j
push_front ff
it2++
push_front yu
push_back i
cout << *it1
it2++
cout << *it2
cout << *it2
++it1
push_front xc
insert it2 yw
it1--
push_back mrjl
it1++
it1++
it2++
cout << l
it1++
push_front e
push_front ynuk
--it2
*it1 = ir
++it2
*it2 = zv
cout << *it2
it2--
--it2
++it2
push_back d
insert it2 ot
push_back tas
insert it1 mk
push_back yn
it2 = end
push_back vwp
```

```
push_front vdwx
it2--
it1++
insert it2 czie
cout << *it1
cout << *it2
insert it2 zqf
++it2
++it2
cout << *it1
push_back tl
cout << *it1
cout << *it1
push_front en
insert it1 li
push_back ual
cout << *it1
*it1 = v
cout << l
cout << l
push_front wz
it2--
++it1
it2++
++it2
push_front p
push_back ki
--it2
++it2
*it2 = hvy
cout << l
it2 = erase it2
it1++
cout << *it1
it1--
cout << *it1
it2--
++it1
*it1 = lnel
insert it1 mm
--it1
it2--
--it2
--it2
push_front cf
push_back ilaw
++it2
push_back esz
push_front wv
--it2
cout << *it2
--it1
insert it2 zi
++it1
--it1
cout << *it1
cout << *it1
insert it1 i
it2++
cout << l
cout << l
```

```
cout << *it1
pop_back
--it2
push_back sq
push_back r
cout << l
insert it2 i
insert it1 bzt
it2++
*it1 = zoay
push_back cc
it2 = erase it2
++it2
it2++
cout << *it1
it1++
it1--
cout << *it1
it2--
push_back rj
++it1
cout << *it1
push_back wku
push_back ura
--it2
push_front secy
insert it2 t
it2++
--it2
it2++
--it1
insert it2 ovn
it1 = begin
--it2
cout << *it1
--it2
push_front yfpn
it1++
insert it2 uc
insert it1 b
pop_front
--it2
insert it2 yx
--it1
it1 = begin
cout << *it2
push_back sasc
push_back q
cout << *it2
it2++
++it1
*it2 = g
push_front hdvm
pop_back
it2 = begin
it2--
*it2 = mr
*it2 = i
it2++
--it2
it1++
```

```
insert it1 tky
push_back j
it2 = erase it2
push_front r
it2--
push_front uio
push_front pnxv
++it1
cout << *it2
cout << l
*it1 = h
insert it2 lte
push_front mo
insert it2 g
it1--
--it2
insert it1 jrh
it1--
cout << *it1
cout << *it2
it2--
--it1
push_front keb
it1--
cout << *it1
push_front zp
cout << *it2
cout << l
```

Exemple de sortida 2

```
bi
bi
pgjh
p y ssg rpqe am tj dec lf bi yqs t p e [pgjh] flh lmfd e (rq)
rq
pgjh
rq
d a rrg g g a p y ssg rpqe am tj dec lf bi yqs t p e gfou wp wrl flh lmfd e rq jnec ac dlx ([])
zq d a rrg g g a p y ssg rpqe am tj dec lf bi yqs t p e gfou wp wrl flh lmfd e rq jnec ac dlx [x
dlx
dlx
xy
pkfc
pkfc
qivz
vvv
mgm
mgm
mgm
nma
e
j
j
bgsb
bgsb
mzrn
mgm
mgm
j
j
xc yu ff j fg s iksl vb fw mkru e mgm mzrn ([yw]) j av bgsb j s gln zq d a rrg g g a p y ssg rpq
```

```
zv
av
vwp
av
av
av
av
en vdwx ynuk e xc yu ff j fg s iksl vb fw mkru e mgm mz
en vdwx ynuk e xc yu ff j fg s iksl vb fw mkru e mgm mz
p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mkru e m
j
bgsb
tl
bgsb
bgsb
wv cf p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mk
wv cf p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mk
bgsb
wv cf p wz en vdwx ynuk e xc yu ff j fg s iksl vb fw mk
zoay
zoay
mm
secy
uc
uc
b
pnxv uio r secy [b] tky wv (cf) p wz en vdwx ynuk e xc
jrh
tky
b
jrh
zp keb mo pnxv uio r secy lte g (b) tky [jrh] wv h p wz
```

Observació

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

- Solució lenta: 5 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 : 2024-01-11 13:58:43

Afegir el mètode push_front a la classe Queue — X94369_ca

Implementeu un nou mètode de la classe Queue, anomenat `push_front`, que permeti afegir un element al principi de la cua, enlloc de al final com passa amb el mètode `push` ja existent. En cas de dubte, mireu el següent programa i el seu comportament descrit en els seus comentaris.

```
Queue<int> q;               // q:
q.push(1);                  // q: 1
q.push(2);                  // q: 1,2
q.push(3);                  // q: 1,2,3
cout << q.front() << endl; // escriu 1
q.push_front(4);            // q: 4,1,2,3
cout << q.front() << endl; // escriu 4
q.pop()                     // q: 1,2,3
cout << q.front() << endl; // escriu 1
```

D'entre els fitxers que s'adjunten en aquest exercici, trobareu `queue.hh`, a on hi ha una implementació de la classe genèrica `Queue`. Busqueu dins `queue.hh` les següents línies i implementeu el mètode:

```
// Pre:
// Post: value ha estat afegit al principi de la cua representada pel paràmet
//void push_front(T value) {
//}
```

D'entre els fitxers que s'adjunten a l'exercici també hi ha `main.cc` (programa principal), i el podeu compilar directament, doncs inclou `queue.hh`. Només cal que pugeu `queue.hh` al jutge.

Entrada

La entrada del programa és una seqüència d'instruccions del següent tipus que s'aniran aplicant sobre una cua d'strings que se suposa inicialment buida:

```
push x (x és string)
pop
front
size
push_front x (x és string)
```

Se suposa que la seqüència d'entrada serà correcta (sense pop ni front sobre cua buida).
El programa principal que us oferim ja s'encarrega de llegir aquestes entrades i fer les crides als corresponents mètodes de la classe cua. Només cal que implementeu el mètode abans esmentat.

Sortida

Per a cada instrucció `front`, s'escriurà el front actual de la cua. Per a cada instrucció `size`, s'escriurà la mida de la cua. El programa que us oferim ja fa això. Només cal que implementeu el mètode abans esmentat.

Exemple d'entrada 1

```
size
size
push_front a
front
size
push_front b
front
size
pop
front
size
pop
size
push c
front
size
push d
front
size
push_front e
front
size
front
size
push f
front
size
pop
front
size
front
size
pop
front
size
pop
front
size
front
size
front
size
pop
size
size
```

Exemple de sortida 1

```
0
0
a
1
b
2
a
1
0
c
1
c
2
e
3
e
3
e
4
c
3
c
3
d
2
f
1
f
1
f
1
0
0
```

Exemple d'entrada 2

```
push rb
front
size
push b
front
size
push ar
front
size
push k
front
```

```
size
pop
front
size
pop
front
size
push dq
front
size
push xr
front
size
```

```
push w
front
size
push_front sj
front
size
push_front db
front
size
pop
front
size
push r
front
size
push_front n
front
size
pop
front
size
pop
front
size
push g
front
size
push_front kl
front
size
pop
front
size
push ln
front
size
pop
front
size
push qf
front
size
push op
front
size
pop
front
size
pop
front
size
push h
front
size
push ku
front
size
pop
front
size
pop
front
size
push m
front
size
push_front uq
front
size
push_front ji
front
size
push m
front
size
push t
front
size
push_front x
front
size
```

Exemple de sortida 2

```
rb
1
rb
2
rb
3
rb
4
b
3
ar
2
ar
3
ar
4
ar
5
sj
6
db
7
sj
6
sj
7
n
8
sj
7
ar
6
ar
7
```

```
kl
8
ar
7
ar
8
k
7
k
8
k
9
dq
8
xr
7
xr
8
xr
9
w
8
r
7
r
8
uq
9
ji
10
ji
11
ji
12
x
13
```

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**, 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 : 2024-01-11 11:53:47