Suma i mida de molts arbres (copy) X32880_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,...,tn`, 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

```
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
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
```

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 : 2023-11-21 01:17:58