Conjunt de valors d'un arbre binari a certa profunditat S74091_ca

Definim la profunditat d'un node en un arbre com:

$$\begin{cases} 1 & \text{si és l'arrel (no té pare); i,} \\ p + 1 & \text{si el node pare té profunditat } p. \end{cases}$$

Per exemple, un arbre binari a on tots els nodes tenen com a valor la seva pròpia profunditat seria:

```
1
|-- 2
|   |-- 3
|   '-- #
'-- 2
    |-- 3
    |   |-- #
    |   '-- 4
    |       |-- #
    |       '-- 5
    '-- 3
```

Implementeu la següent funció:

```
/**
 * @brief Retorna el conjunt dels valors dels nodes d'un arbre
 *        binari que estan a la profunditat `depth`
 *
 * @param  t      Arbre binari d'enters.
 * @param  depth  La profunditat dels nodes que es vol.
 *
 * @returns El conjunt de valors trobats a profunditat `depth`
 */
set<int> values_at_depth(BinTree<int> t, int depth);
```

Entrada

Cada cas consisteix en una representació textual d'un arbre binari d'enters i un enter que és una profunditat. (Aquesta lectura ja la fa el programa principal.)

Sortida

Per a cada cas, es mostra el conjunt de valors dels nodes, amb els valors ordenats, separats per espais, i delimitats per "{" i "}". (La sortida també la fa el programa principal.)

Observació

Els fitxers públics (icona del gatet) contenen:

`bintree.hh`	la classe `BinTree`
`bintree-io.hh`	l'entrada/sortida de `BinTree` (format "visual")
`bintree-inline.hh`	l'entrada/sortida de `BinTree` (format "inline")
`main.cc`	el programa principal
`Makefile`	per compilar amb `make` còmodament
`.vscode`	configuració per compilar i debuggar amb VSCode

Cal d'implementar `values_at_depth` en un **fitxer `.cc` nou**, compilar (està preparat per poder compilar i debuggar amb VSCode), i finalment **enviar només el fitxer amb la funció**.

Exemple d'entrada 1

```
visual
3
|-- 6
'-- 6

2

1
|-- 4
|   |-- #
|   '-- 2
'-- 5
    |-- 8
    '-- #

3

8
|-- 9
|   |-- 4
|   |   |-- 9
|   |   '-- 7
|   '-- 5
|       |-- 1
|       '-- 1
'-- #

4
```

Exemple de sortida 1

```
{ 6 }
{ 2 8 }
{ 1 7 9 }
```

Exemple d'entrada 2

```
inline
22(12(12(27,25),30(31,37)),30(23(9,18),34(37,30)))
1
32(21(36(40,37),24(16,31)),37(29(27,5),31(36,5)))
2
4(22(7(31,39),19(13,8)),32(5(9,34),20(25,30)))
4
10(19(1(28,33),5(3,39)),38(32(37,11),28(29,16)))
4
37(26(20(3,30),4(20,8)),18(34(36,4),39(3,18)))
3
10(14(11(16,10),12(23,28)),13(18(38,32),21(36,39)))
2
25(19(34(14,32),22(21,25)),21(3(26,4),1(38,28)))
2
1(26(36(4,3),19(39,28)),27(18(29,15),39(13,11)))
4
30(33(38(21,14),6(31,19)),16(6(11,37),14(27,30)))
3
20(23(3(5,35),25(4,14)),38(27(6,6),23(33,38)))
1
```

Exemple de sortida 2

```
{ 22 }
{ 21 37 }
{ 8 9 13 25 30 31 34 39 }
{ 3 11 16 28 29 33 37 39 }
```

```
{ 4 20 34 39 }
{ 13 14 }
{ 19 21 }
{ 3 4 11 13 15 28 29 39 }
{ 6 14 38 }
{ 20 }
```

Informació del problema

Autor : Pau Fernández
Generació : 2025-03-30 18:49:08