Expressió. Representació infixa amb el màxim nombre de parèntesis X21901_ca

Donada la classe *expressio* que permet guardar expressions matemàtiques en un arbre binari usant memòria dinàmica, cal implementar el mètode

```
string llista_tokens_parentitzada () const;
```

que retorna un string amb la representació infixa de l'expressió amb tots els parèntesis possibles, excepte quan són operands (constants o variables) que mai fan falta. Els operadors i operands es guarden en l'string token de cada node. Els operadors unaris (+ - sqrt log exp) tenen el fill dret buit.
Cal enviar a jutge.org la següent especificació de la classe *expressio* i la implementació del mètode dins del mateix fitxer.

```
#include <cstdlib>
#include <iostream>
using namespace std;
typedef unsigned int nat;

class expressio {
  // Guarda una expressio en un arbre binari.
  // Els operadors i operands es guarden en l'string token de cada node.
  // Els operadors unaris (+ - sqrt log exp) tenen el fill dret buit.
  public:
    expressio (): _arrel (nullptr) {};
    // Pre: cert
    // Post: el resultat és un arbre sense cap element
    expressio (expressio &ae, const string &x, expressio &ad);
    // Pre: cert
    // Post: el resultat és un arbre amb un element i dos subarbres

    // Les tres grans
    expressio (const expressio &a);
    ~expressio ();
    expressio & operator=(const expressio& a);

    // operador « d'escriptura
    friend std :: ostream& operator<<(std::ostream&, const expressio &a);

    // operador » de lectura
    friend std :: istream& operator>>(std::istream&, expressio &a);

    string llista_tokens_parentitzada () const;
    // Pre: Cert
    // Post: Retorna string amb la representació infixa de l'expressió amb tots els parèntesis possibles,
    // excepte quan són operands (constants o variables) que mai fan falta.
```

```
  private:
    struct node {
      node* f_esq;
      node* f_dret;
      string token;
    };
    node* _arrel;
    static node* copia_nodes(node* m);
    static void esborra_nodes(node* m);
    static void print_nodes(node* m, ostream &os, string d1);

    // Aquí va l'especificació dels mètodes privats addicionals
};

// Aquí va la implementació del mètode llista_tokens_parentitzada i privats addicionals
```

Per testejar la solució, jutge.org ja té implementats la resta de mètodes de la classe *expressio* i un programa principal que llegeix una expressió i després crida el mètode *llista_tokens_parentitzada*.

Entrada

L'entrada consisteix en la descripció de l'arbre de l'expressió (el seu recorregut en preordre, en el qual inclou les fulles marcades amb l'string "#"). Per exemple, l'arbre

```
[*]
 \__[pt]
 |    \__#
 |    \__#
 \__[2]
      \__#
      \__#
```

es descriuria amb

```
* 2 # # pt # #
```

Sortida

El contingut de l'arbre binari seguit per l'string que retorna el mètode *llista_tokens_parentitzada*.

Observació

Només cal enviar la classe requerida i la implementació del mètode *llista_tokens_parentitzada*. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.

Exemple d'entrada 1

```
34 # #
```

Exemple de sortida 1

```
[34]
 \__#
```

```
 \__#
```

```
34
```

Exemple d'entrada 2

```
x # #
```

Exemple de sortida 2

```
[x]
 \__#
 \__#

x
```

Exemple d'entrada 3

```
log 3 # # #
```

Exemple de sortida 3

```
[log]
 \__#
 \__[3]
     \__#
     \__#

log(3)
```

Exemple d'entrada 4

```
* 2 # # pt # #
```

Exemple de sortida 4

```
[*]
 \__[pt]
 |    \__#
 |    \__#
 \__[2]
     \__#
     \__#

(2*pt)
```

Exemple d'entrada 5

```
+ 2 # # * 3 # # y # #
```

Exemple de sortida 5

```
[+]
 \__[*]
 |    \__[y]
 |    |    \__#
 |    |    \__#
 |    \__[3]
 |         \__#
 |         \__#
 \__[2]
     \__#
     \__#

(2+(3*y))
```

Exemple d'entrada 6

```
+ sqrt 2 # # # * 3 # # y # #
```

Exemple de sortida 6

```
[+]
 \__[*]
 |    \__[y]
 |    |    \__#
 |    |    \__#
 |    \__[3]
 |         \__#
 |         \__#
 \__[sqrt]
     \__#
```

```
      \__[2]
          \__#
          \__#
```

```
(sqrt(2)+(3*y))
```

Exemple d'entrada 7

```
+ sqrt 2 # # # * - 3 # # # y # #
```

Exemple de sortida 7

```
[+]
 \__[*]
 |    \__[y]
 |    |    \__#
 |    |    \__#
 |    \__[-]
 |         \__#
 |         \__[3]
 |              \__#
 |              \__#
 \__[sqrt]
      \__#
      \__[2]
           \__#
           \__#
```

```
(sqrt(2)+(-(3)*y))
```

Exemple d'entrada 8

```
log * - z # # # ^ x # # / + sqrt y # # # 7[1]#og#]3 # # #
```

Exemple de sortida 8

```
  \__#
  \__[*]
       \__[^]
       |    \__[/]
       |    |    \__[3]
       |    |    |    \__#
       |    |    |    \__#
       |    |    \__[+]
       |    |         \__[7]
       |    |         |    \__#
       |    |         |    \__#
       |    |         \__[sqrt]
       |    |              \__#
       |    |              \__[y]
       |    |                   \__#
       |    |                   \__#
       |    \__[x]
       |         \__#
       |         \__#
       \__[-]
            \__#
            \__[z]
                 \__#
                 \__#
```

```
log((-(z)*(x^((sqrt(y)+7)/3))))
```

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T14:22:20.112Z

© *Jutge.org*, 2006–2026.
https://jutge.org