Heap amb memòria dinàmica. Eliminar mínim — X14777_ca

Donada la classe *heap* que permet gestionar Min-Heaps usant memòria dinàmica, cal implementar el mètode

void *elim_min* ();

que, en un Min-Heap no buit, elimina l'element més petit del Min-Heap o qualsevol d'ells si està repetit.
La classe *heap* guarda els elements del Min-Heap en un arbre binari implementat amb memòria dinàmica. Conté dos atributs que guarden el nombre d'elements del Heap (*_nelems*) i el punter al node arrel de l'arbre binari (*_arrel*). Cada node de l'arbre binari conté l'element (*info*) i 3 punters: al fill esquerre (*fesq*), al fill dret (*fdret*) i al pare (*pare*).
Cal enviar a jutge.org la següent especificació de la classe *heap* i la implementació del mètode dins del mateix fitxer. Indica dins d'un comentari a la capçalera del mètode el seu cost en funció del nombre d'elements del Min-Heap.

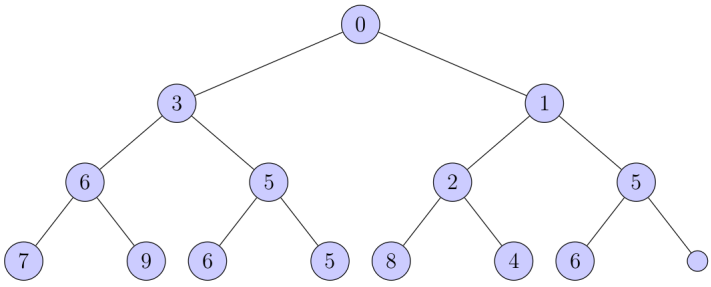
```
#include <cstdlib>
using namespace std;
typedef unsigned int nat;

template <typename T>
class heap {
  public:
    heap (): _arrel (NULL), _nelems(0) {};
    // Pre: Cert
    // Post: El resultat és un heap sense cap element

    heap(heap<T> &he, const T &x, heap<T> &hd);
    // Pre: Cert
    // Post: El resultat és un heap amb x com arrel, he com a fill
    // esquerre i hd com a fill dret. No fa còpia dels heaps he i hd

    heap(const heap<T> &h);
    // Pre: Cert
    // Post: Crea un heap que és una còpia de h

    ~heap ();
    // Pre: Cert
    // Post: El heap ha estat destruït

    bool empty() const;
    // Pre: Cert
    // Post: Retorna un booleà indicant si el heap està buit

    T min() const;
    // Pre: El heap no és buit
    // Post: Retorna l'element mínim del heap
```

```
    void elim_min();
    // Pre: El heap no és buit
    // Post: S'ha eliminat l'element mínim del heap o qualsevol d'ells si està repetit

  private:
    struct node {
      node* fesq;   // Punter al fill esquerre
      node* fdret;  // Punter al fill dret
      node* pare;   // Punter al pare
      T info;
    };
    node* _arrel;  // Arrel de l'arbre binari del heap
    int _nelems;   // Nombre d'elements del heap

    static node* copia_nodes(node* m, node *n_pare);
    static void esborra_nodes(node* m);

    void ultim(node* &ult, node* &pare_ult) const;
    // Pre: Cert
    // Post: ult conté el punter de l'últim element del heap (NULL si està buit)
    // pare_ult conté el punter del pare de l'últim element del heap (NULL si no en té)

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

// Aquí va la implementació del mètode elim_min
```

Per testejar la solució, jutge.org ja té implementats la resta de mètodes de la classe *heap* i un programa principal que llegeix un Min-Heap i després crida vàries vegades els mètodes *min* i *elim_min* fins que el Min-Heap queda buit.

Entrada

L'entrada consisteix en la descripció dels elements que formen l'arbre binari d'un Min-Heap d'enters (el nombre d'elements seguit pel seu recorregut en preordre). Per exemple, l'arbre (mira el PDF de l'enunciat)



es descriuria amb

```
14
0 3 6 7 9 5 6 5 1 2 8 4 5 6
```

Sortida

Una línia amb tots els elements del Min-Heap obtinguts al cridar els mètodes *min* i *elim_min* alternativament fins que el Min-Heap queda buit. Cada element està separat amb un espai.

Observació

Només cal enviar la classe requerida i la implementació del mètode *elim_min*. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.
Indica dins d'un comentari a la capçalera del mètode el seu cost en funció del nombre d'elements del Min-Heap.
Et pot ser de molta utilitat el mètode privat *ultim* que calcula aquests dos punters:

- *ult*: punter de l'últim element del heap (NULL si està buit).
- *pare_ult*: punter del pare de l'últim element del heap (NULL si no en té).

Exemple d'entrada 1

```
1
3
```

Exemple de sortida 1

```
3
```

Exemple d'entrada 2

```
2
3 5
```

Exemple de sortida 2

```
3 5
```

Exemple d'entrada 3

```
3
3 5 4
```

Exemple de sortida 3

```
3 4 5
```

Exemple d'entrada 4

```
0
```

Exemple de sortida 4

```

```

Exemple d'entrada 5

```
14
0 3 6 7 9 5 6 5 1 2 8 4 5 6
```

Exemple de sortida 5

```
0 1 2 3 4 5 5 5 6 6 6 7 8 9
```

Exemple d'entrada 6

```
11
0 3 6 7 9 5 6 5 1 2 5
```

Exemple de sortida 6

```
0 1 2 3 5 5 5 6 6 7 9
```

Informació del problema

Autor : Jordi Esteve
Generació : 2021-01-24 19:28:22