HotSet Y89872_es

En este ejercicio implementaréis una nueva clase de datos, llamada `HotSet`. Los objetos de esta clase representan conjuntos de elementos de un cierto tipo `T` sin repeticiones, ordenados por orden de consulta o de inserción. La idea de esta estructura es que los elementos más frecuentemente consultados o insertados queden al inicio de la estructura, y por tanto se reduzca mucho el coste de acceder a ellos, dado que la búsqueda es lineal.

El caso de uso de un `HotSet` son los problemas donde la probabilidad de volver a utilizar un elemento del `HotSet` que se ha usado recientemente es mucho más alta que la probabilidad de usar otros elementos.

La clase se implementa con una lista simplemente encadenada: los elementos tienen la dirección del siguiente, pero no del anterior. La estructura también tiene un centinela, de nombre `cent`, que es un elemento que marca el inicio, pero no tiene contenido.

La plantilla del fichero `hot_set.hh` para hacer la implementación es la siguiente:

```
#ifndef HOT_SET_HH
#define HOT_SET_HH

#include <iostream>
#include <string>
using namespace std;

template <typename T>
class HotSet {
    public:
    HotSet() {
        tam = 0;
        cent = new nodo;
        cent->sig = nullptr;
    }

    /// >>>> Implementar aquí los métodos públicos <<<<

 private:
    struct nodo {
        T info;
        nodo *sig; // apuntador al siguiente elemento
    };

    nodo *cent;  // apuntador al centinela, que *no* es propiamente un elemento
    int tam;

    /// >>>> Implementar los métodos privados necesarios <<<<
};

#endif  // HOT_SET_HH
```

Así pues:

- un `HotSet` que está vacío siempre tendrá el `cent`inela, y se cumple que `cent->sig == nullptr;`
- si un `HotSet` tiene uno o más elementos, `cent->sig` apunta al primer elemento; y,
- el último elemento de un `HotSet` no tiene sucesores, es decir, si `ult` es un apuntador al último elemento, entonces `ult->sig == nullptr`.

Se trata, entonces, de implementar los métodos que faltan de la clase `HotSet`, de tal manera que el programa principal (sin modificar) funcione correctamente.

Entrada

El programa principal (y también el `Makefile`) ya se proporciona, en el icono del gatito, y lee una secuencia de comandos que ejercitan las operaciones de la clase `HotSet`. Cada comando es una línea que comienza con una palabra y llama a algun método del `HotSet`.

Salida

La salida también está implementada en el programa principal. Cada vez que se llama a un método de la clase `HotSet`, se muestra algun mensaje en la salida estándar.

Observación

Enviad un `.tar` tal como el que podéis descargar en el icono del gatito, añadiendo el fichero `hot_set.hh` con vuestra implementación.

Ejemplo de entrada 1

```
ins 1
ins 2
front
ins 3
has 1
front
size
has 3
show
```

Ejemplo de salida 1

```
primero: 2
1 pertenece
primero: 1
3 elementos
3 pertenece
3 1 2
```

Ejemplo de entrada 2

```
ins 1
ins 2
ins 3
show
ins 2
show
has 5
has 3
front
ins 1
show
```

Ejemplo de salida 2

```
3 2 1
2 3 1
5 no pertenece
3 pertenece
primero: 3
1 3 2
```

Información del problema

Autoría: PRO2

Generación: 2026-01-25T21:39:11.073Z