Mueve al principio en una lista V39031_es

Implementad un nuevo método de la clase `List` para mover el contenido apuntado por un iterador al inicio de la lista. En caso de que el iterador ya apunte al primer elemento de la lista, entonces el método no hará nada. Entre los archivos que se adjuntan en este ejercicio, encontraréis `list.hh`, donde hay una implementación de la clase genérica List.

```
/**
 * @pre  'it' apunta a algún elemento de la lista implícita no vacía.
 *
 * @post 'it' continúa apuntando al mismo elemento, el cual ha sido movido
 *       al inicio de la llista. No se ha reservado ni liberado memoria.
 *       En el caso en que el elemento apuntado por 'it' ya fuese el
 *       primero, nada ha cambiado.
 */
void moveToBeginning(iterator &it)
```

(Encontraréis la declaración de `moveToBeginning` al final del fichero `list.hh`, que es bastante largo.)

No toquéis el resto de la implementación de la clase `List`, excepto si, por algún motivo, consideráis que necesitáis añadir algún método auxiliar a la parte privada. Entre los archivos que se adjuntan al ejercicio (icono del gatito) está el programa principal, `main.cc`, que podéis compilar.

Los ficheros públicos (icono del gatito) incluyen un `.tar` con `main.cc`, `list.hh` y un `Makefile`. También se incluye una copia de los juegos de prueba públicos por comodidad. En la carpeta donde se descompriman se puede: compilar con "`make`"; y testear con "`make test`".

El `main.cc` ya se encarga de leer la entrada, procesar los comandos y producir la salida. Para entregar, solo hay que subir al Juez vuestro fichero `list.hh` modificado.

Entrada

La entrada del programa es una secuencia de instrucciones del siguiente tipo que se irán aplicando sobre una lista que inicialmente está vacía y un iterador que está situado inicialmente al principio (y final) de esta lista:

```
push_front s     // s es un string
push_back s      // s es un string
pop_front
pop_back
it++
it--
*it
moveToBeginning  // el método que debéis implementar
print_list       // muestra la lista entera en la salida
```

Se puede suponer que la secuencia de entrada será correcta (es decir: sin `pop_front` ni `pop_back` sobre lista vacía, ni `*it` ni `moveToBeginning` teniendo `it` situado en el `end` de la lista; tampoco habrá `pop_front` justo cuando el iterador esté apuntando al primer elemento de la lista, ni habrá `pop_back` justo cuando el iterador esté apuntando al último elemento de la lista; tened en cuenta que el último elemento de la lista no es el `end` de la lista).

El programa principal que os ofrecemos ya se encarga de leer estas entradas y hacer las llamadas a los correspondientes métodos de la clase `List`. Solo tenéis que implementar el método antes mencionado.

Salida

El programa principal ya se encarga de producir la salida: para cada instrucción `*it`, escribe el contenido apuntado por el iterador; y para cada instrucción `print_list` se muestra el contenido de toda la lista.

Ejemplo de entrada 1

```
push_back a
it--
*it
print_list
moveToBeginning
print_list
push_back b
it++
*it
print_list
moveToBeginning
print_list
push_front c
it++
*it
print_list
moveToBeginning
print_list
```

Ejemplo de salida 1

```
a
a
a
b
a b
b a
a
c b a
a c b
```

Información del problema

Autoría: PRO2

Generación: 2026-01-27T18:49:36.901Z