## Deque V79239_es

En este ejercicio completaremos la implementación de una estructura de datos llamada `Deque`. La librería estándar de C++ contiene una clase `deque` pero la implementación de este ejercicio es una simplificación.

Un `Deque` (*double-ended queue*) funciona como una pila y una cola a la vez. Podemos introducir elementos al inicio (`push_front`) y al final (`push_back`), y eliminar elementos del inicio (`pop_front`) y del final (`pop_back`). En un `Deque`, el método `push_back` se comporta exactamente como `push` en una cola clásica; y `pop_front` se comporta como el `pop` en una cola clásica.

Con el fin de poder aprovechar la implementación de `Queue` y simplemente añadir operaciones nuevas, hemos usado la clase `Queue` de base cambiando la implementación de `Item`, que ahora también tiene un apuntador al elemento anterior. Es decir, la estructura de nodos es equivalente a la de `Queue`, pero está *doblemente enlazada*.

Familiarízate con la clase `Deque<T>`, y luego implementa un *método público* que elimine el último elemento del Deque (`pop_back`). La declaración es la siguiente:

```
/**
 * @brief  El Deque original (p.i.) pierde el último elemento si no está vacío.
 *         si está vacío se aborta la ejecución y se envía un mensaje de error.
 *
 * @pre   El Deque original (p.i.) no está vacío.
 * @post  El Deque pierde el último elemento.
 */
void pop_back();
```

### Observación

Para poder evaluar el uso de punteros, *no uséis otros métodos, ni públicos ni privados* de la clase para resolver el problema, acceded siempre a los miembros privados directamente.

Si tenéis dudas sobre cómo producir el mensaje de error, mirad cómo se hace en las operaciones que ya damos implementadas.

Los ficheros públicos (icono del gatito) contienen:

| | |
|---|---|
| `deque.hh` | la clase `Deque<T>` |
| `main.cc` | el programa principal (gestiona la entrada y salida) |
| `Makefile` | para compilar con `make` en el terminal |
| `.vscode` | para compilar y depurar con F5 |

Para entregar solo hay que **enviar el fichero `deque.hh` modificado.**

### Entrada

*De la entrada ya se encarga el programa principal.* La entrada está formada por diferentes operaciones sobre el Deque: `push_back`, `push_front`, `pop_back`, `pop_front`, `size`, `front`, `back`, `write_deque`, acabadas con `end`.

## Salida

*De la salida también se encarga el programa principal.* La salida muestra los resultados de las operaciones de consulta y el estado del Deque cuando se solicita.

### Ejemplo de entrada 1

```
push_back 1
push_back 2
push_back 3
pop_back
size
write_deque
push_back 10
push_back 20
pop_back
pop_back
size
write_deque
push_back 5
pop_back
size
write_deque
end
```

### Ejemplo de salida 1

```
2
 1 2
2
 1 2
2
 1 2
```

### Ejemplo de entrada 2

```
push_front 1
size
write_deque
push_front 2
push_front 3
size
write_deque
front
back
push_front 10
push_front 20
size
write_deque
end
```

### Ejemplo de salida 2

```
1
 1
3
 3 2 1
3
1
5
 20 10 3 2 1
```

## Información del problema

Autoría: Mª Lluïsa Bonet i Pau Fernández

Generación: 2026-03-25T18:26:14.291Z