Deque V79239_ca

En aquest exercici cal completar la implementació d'una estructura de dades anomenada `Deque<T>`. (La llibreria estàndard de C++ conté una classe `deque` però la implementació que fem en aquest exercici n'és una simplificació.)

Un `Deque<T>` (*double-ended queue*) funciona com una pila i una cua alhora. Podem introduir elements a l'inici (`push_front`) i també al final (`push_back`), i eliminar elements de l'inici (`pop_front`) i també del final (`pop_back`). Per tant, els mètodes `push` i `pop`, que a la pila i la cua afegeixen i eliminen elements d'un sol extrem, ara han d'indicar de quin extrem es tracta, i n'hi ha dos per a cada costat.

Amb la finalitat de poder aprofitar la implementació de `Queue` i simplement afegir operacions noves, hem usat `queue.hh` de base i hem canviat la implementació de `Item`, que ara també té un apuntador a l'element anterior. És a dir, l'estructura de nodes és equivalent a la de `Queue`, però està *doblement enllaçada*.

Primer familiaritza't amb la classe `Deque<T>`, i després implementa els dos mètodes nous següents:

1. Un *mètode públic* que elimini l'últim element del `Deque<T>` (`pop_back`). La declaració és la següent:

```
/**
 * @brief  El Deque<T> (p.i.) perd l'últim element si no està buit.
 *         Si està buit, s'aborta l'execució i s'envia un
 *         missatge d'error.
 *
 * @pre   El Deque<T> (p.i.) no està buit.
 * @post  El Deque<T> perd l'últim element.
 */
void pop_back();
```

2. Un *mètode públic* que afegeixi un element pel principi (pel "front"), del `Deque<T>` (`push_front`). La declaració és la seguent:

```
/**
 * @brief  Afegeix un element al principi del Deque<T> (el "front").
 *
 * @param x L'element a afegir.
 * @post  El Deque<T> (p.i.) té un element més al principi i el nou
 *        element té el valor `x`.
 */
void push_front(const T& x);
```

Observació

Per poder avaluar l'ús de punters, *no useu altres mètodes, ni públics ni privats* de la classe per resoldre el problema, accediu sempre als membres privats directament.

Si teniu dubtes sobre com produir el missatge d'error, mireu com es fa a les operacions que ja us donem implementades.

Els jocs de prova estan separats en dos grups per poder treballar per separat en cada mètode. Un grup de jocs de prova inclou crides als mètodes existents de `Deque<T>` i també `push_front`; i l'altre grup inclou crides als mètodes de `Deque<T>` i també `pop_back`.

Els fitxers públics (icona del gatet) contenen:

`deque.hh`	la classe `Deque<T>`
`main.cc`	el programa principal (gestiona l'entrada i sortida)
`Makefile`	per compilar amb `make` al terminal
`.vscode`	per compilar i depurar amb F5

Per lliurar només cal **enviar el fitxer `deque.hh` modificat.**

Entrada

De l'entrada ja s'encarrega el programa principal. L'entrada està formada per diferents operacions sobre el `Deque<T>`: `push_back`, `push_front`, `pop_back`, `pop_front`, `size`, `front`, `back`, `write_deque`, acabades amb `end`.

Sortida

De la sortida també s'encarrega el programa principal. La sortida mostra els resultats de les operacions de consulta i l'estat del `Deque<T>` quan se sol·licita.

Exemple d'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
```

Exemple de sortida 1

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

Exemple d'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
```

Exemple de sortida 2

```
1
 1
3
```

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

Informació del problema

Autoria: Mª Lluïsa Bonet i Pau Fernández

Generació: 2026-03-25T18:26:11.452Z