# 12. gyakorlat

It's official – Nincs CodeBlocks vizsgán. Természetesen segédanyagot lehet vinni.

## Adapteres ZH (Stack​Queue) + Megoldás

Ez megy legnehezebben mindenkinek, nem tudni miért.

|  |  |
| --- | --- |
| $$main.cpp$$ | $$stque.h$$ |
| #include "stque.h"#include <list>#include <iostream>#include <algorithm>const int max **=** 10000**;**int main**(){** int yourMark **=** 1**;** // 2-es StackQueue**<**int**,** **true>** st**;** st**.**push**(**3**);** st**.**push**(**4**);** StackQueue**<**double**,** **false>** q**;** q**.**push**(**5.33**);** q**.**push**(**4.78**);** const StackQueue**<**int**,** **true>** cst **=** st**;** **if** **(**4 **==** cst**.**get**()** **&&** 5.33 **==** q**.**get**())** yourMark **=** cst**.**size**();** // 3-as st**.**pop**();** **for(**int i **=** 0**;** i **<** max**;** **++**i**){** st**.**push**(**i**);** st**.**pop**();** **}** **if** **(!**st**.**empty**())** yourMark **=** st**.**get**();** // 4-es **++**st**.**get**();** StackQueue**<**int**,** **true,** std**::**list**<**int**>** **>** lsq**;** lsq**.**push**(**1**);** StackQueue**<**double**,** **false,** std**::**deque**<**double**>** **>** dsq**;** dsq**.**push**(**4.3**);** dsq**.**push**(**8.2**);** **if** **(**2 **==** dsq**.**size**()** **&&** 1 **==** lsq**.**get**())** yourMark **=** st**.**get**();** // 5-os const std**::**stack**<**int**>** si **=** lsq**.**data**();** const std**::**queue**<**double**>** qd **=** dsq**.**data**();** const std**::**stack**<**int**>** csi **=** cst**.**data**();** yourMark **=** si**.**top**()** **+** qd**.**size**()** **+** csi**.**size**();** std**::**cout **<<** "Your mark is " **<<** yourMark **<<** std**::**endl**;** **return** 0**;****}** | #ifndef STACKQUEUE\_\_H#define STACKQUEUE\_\_H#include <deque>#include <queue>#include <stack>template **<**bool cond**,** class T**,** class E**>**struct If**{** **typedef** T Ret**;****};**template **<**class T**,** class E**>**struct If**<false,** T**,** E**>{** **typedef** E Ret**;****};**template **<**class T**,** bool is\_stack**,** class Cont **=** std**::**deque**<**T**>** **>**class StackQueue**{**protected**:** Cont c**;**public**:** void push**(**const T**&** t**){** c**.**push\_back**(**t**);** **}** int size**()** const**{** **return** c**.**size**();** **}** const T**&** get**()** const**{** **if** **(**is\_stack**)** **return** c**.**back**();** **else** **return** c**.**front**();** **}** T**&** get**(){** **if** **(**is\_stack**)** **return** c**.**back**();** **else** **return** c**.**front**();** **}** bool empty**()** const**{** **return** c**.**empty**();** **}** void pop**(){** **if** **(**is\_stack**)** c**.**pop\_back**();** **else** c**.**pop\_front**();** **}** typename If**<**is\_stack**,** std**::**stack**<**T**>,** std**::**queue**<**T**>** **>::**Ret data**()** const**{** typename If**<**is\_stack**,** std**::**stack**<**T**>,** std**::**queue**<**T**>** **>::**Ret d**;** **for(**typename Cont**::**const\_iterator i **=** c**.**begin**();** i**!=**c**.**end**();++**i**)** d**.**push**(\***i**);** **return** d**;** **}****};**#endif |

## Megjegyzések

A $StackQueue$ olyan adatszerkezet, ami hol $Stack$-ként, hol $Queue$-ként tud működni. A template paraméter dönti el, melyik.

Érdemes megnézni a magasabb jegyek követelményeit, ahonnan kiderül, hogy van egy harmadik template paraméter is, ami default értékkel rendelkezik. (A használt konténer, alapból $deque$.)

Kell-e copy konstruktort írni ehhez: const StackQueue**<**int**,** **true>** cst **=** st**;** Nem.

$Stack$ = LIFO "Last in, First out". STL-ben a $pop$ és $top$ különböző művelet. Az egyik kitörli, a másik kiolvassa. Törlés nem adja vissza mit törölt, sajnos.

$Queue$ = FIFO "Fist in, First out". Nekünk ezt a kettő típust kell vegyíteni. A különbség a kettő között minimális.

Erre jó jelölt a $deque$. Tudunk elejére és végére is berakni, kiolvasni onnan.

Lehet a $get$-et így is írni: **return** is\_stack **?** c**.**back**()** **:** c**.**front**();**

**++**st**.**get**();** ⇒ Ez csak azt jelenti, hogy referencia szerint működő $get$-et is kell írni. Hogy tudjuk a $StackQueue$-ben levő jegyünket növelni. Nem kell **operator++**.

Metaprogramozás… typename If**<**is\_stack**,** std**::**stack**<**T**>,** std**::**queue**<**T**>** **>::**Ret data**()** const**{**

Ha is\_stack hamis, akkor std**::**queue**<**T**>**, különben std**::**stack**<**T**>**.

$Multiset$-es példa is ki lett osztva, én nem kértem, mert neten fent van.

Gyakorlat vége.