Kruhový buffer

Obtiažnosť: vysoká

Popis

Cirkulárny buffer, cyklický buffer, resp. kruhový buffer je údajová štruktúra, ktorá používa jeden buffer (pole) s fixnou veľkosťou. Tento buffer pri fungovaní pôsobí ako keby bol spojený do kruhu (t.j. ako keby za jeho koncom nasledoval opäť jeho začiatok).

Kruhový buffer na začiatku začína prázdny a má preddefinovanú veľkosť. Napríklad toto je 7-prvkový buffer:

```
[ ][ ][ ][ ][ ][ ][ ]
```

Predstavme si, že do buffra chceme zapísať hodnotu 1. Pri kruhovom buffri nezáleží na tom, na ktoré miesto v buffri budeme zapisovať prvý prvok:

```
[ ][ ][1][ ][ ][ ][ ]
```

Predstavme si, že potom pridáme ďalšie 2 prvky - 2 a 3. Tieto prvky sa pridávajú za naposledy pridaný prvok, t.j. za prvok 1:

```
[ ][ ][1][2][3][ ][ ]
```

Pri odoberaní prvkov z buffra sa vždy odoberajú najstaršie prvky. Po odobraní 2 prvkov teda bude buffer vyzerať nasledovne:

```
[ ][ ][ ][ ][3][ ][ ]
```

Keď je v buffri 7 prvkov, tak je plný:

```
[6][7][8][9][3][4][5]
```

Keď je buffer plný, tak pri ďalšom pokuse o zápis do neho sa zobrazí chybová hláška, ktorá používateľa upozorní na to, že najskôr musí počkať, kým sa v buffri uvoľní miesto.
Používateľ však môže zvoliť možnosť, aby sa v takejto situácii natvrdo prepísali najstaršie dáta. Takže napríklad ak v našom príklade pridáme natvrdo prvky A a B, prepíšu sa prvky 3 a 4:

```
[6][7][8][9][A][B][5]
```

Po premazaní je nastarším prvkom v buffri prvok 5. Takže ak teraz odoberieme 2 prvky, z buffra vypadnú prvky 5 a 6:

```
[ ][7][8][9][A][B][ ]
```

Nakoľko sa v buffri uvoľnilo miesto, je možné do neho vložiť prvky C a D na voľné miesto, kde predtým boli prvky 5 a 6, nie je potrebné prepisovať prvky 7 a 8. Po vložení prvkov C a D je buffer opäť plný:

```
[D][7][8][9][A][B][C]
```

Technologické požiadavky

Vašou úlohou je vytvoriť triedu CircularBuffer, ktorá bude obsahovať nasledovné členy:

- **privátne** pole znakov s fixnou dĺžkou, do ktorého sa budú ukladať prvky buffra
- metódu **bool Push(char element)** - pokúsi sa zapísať nový prvok na najbližšie voľné miesto v buffri. Ak nie je v buffri dostatočne voľné miesto, vráti sa False. Pri úspešnom vložení sa vráti True.
- metódu **bool ForcePush(char data)** - pokúsi sa zapísať nový prvok do buffra nezávisle na tom, či pri tom prepíše najstarší prvok. Vráti úspešnosť zápisu prvkov.
- metódu **char Pop()** - odoberie z buffra najstarší prvok a vráti ho ako návratovú hodnotu

Pri spustení programu používateľ zadá veľkosť buffra.

Ukážka požadovaného výstupu

Viď. popis.