Zásobník, fronta, seznam

Z FAV wiki

Všechny tyto ADT implementují IsEmpty (prázdná struktura) a Size (pošet prvků ve struktuře)

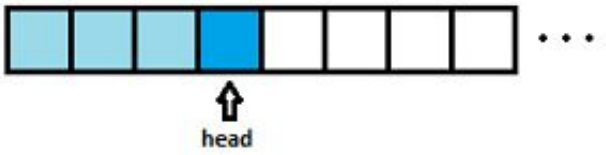
Zásobník

Interface: operace Push, Pop, Peek

- **Push** vkládá data do zásobníku za sebe
- **Pop** vybírá nejpozději vložená data
- **Peek** vrátí nejpozději vložená data bez výběru prvku ze struktury

= LIFO pamět (Last In First Out)

Všechny operace jsou O(1)



Head: vrchol zásobníku

Implementace objektově:

- Prvky (objekty) obsahují data a ukazují na nižší prvky v zásobníku (na prvky pod sebou)
- **Konstruktor:** Vytvoříme ukazatel Head, který nastavíme na null
- **Modifikátor:** Push vytvoří pro data nový prvek, který ukazuje na prvek Head (Head bude předchozí), a změní ukazatel Head na vytvořený prvek
- **Selektor:** Peek vrátí data prvku Head, pokud Head != null. Pop vrátí stejná data jako Peek, ale před návratem změní ukazatel Head na předchozí prvek (Head = Head.Previous)

Implementace na poli:

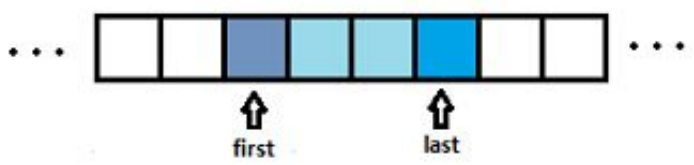
- Máme pole typu stejného jako data
- **Konstruktor:** Alokace pole a indexu Head, nastavení Head na -1
- **Modifikátor:** Push zvýší index Head o 1 a zapíše data do pole na tento index. Pokud je index mimo pole, zvětšení pole (např. 2x), a po té zápis
- **Selektor:** Peek vrátí prvek pole pod indexem Head, pokud Head >= 0. Pop jako peek, ale před návratem provede Head--, pokud Head >= 0.

Fronta

Interface: operace Push, Pop, Front

- **Push** vkládá data do fronty za sebe
- **Pop** vybírá nejdříve vložená data
- **Peek** vrátí nejdříve vložená data bez výběru prvku ze struktury

= FIFO pamět (First In First Out)



First: začátek fronty; Last: konec fronty

Implementace objektově:

- Prvky obsahují data a ukazují na následující prvky fronty
- **Konstruktor:** Vytvoříme pouze ukazatele First a Last, ukazující na null.
- **Modifikátor:** Push vytvoří pro data nový prvek, upraví prvek Last (pokud != null) tak, aby ukazoval na tento prvek, a změní ukazatel Last na tento prvek. Je-li ukazatel First nastaven na null, nastaví tento také na vytvořený prvek.
- **Selektor:** Front vrátí data prvku First, pokud First != null. Pop vrátí stejná data jako Front, ale před návratem změní ukazatel First na následující prvek, pokud First != null (First = First.Next)

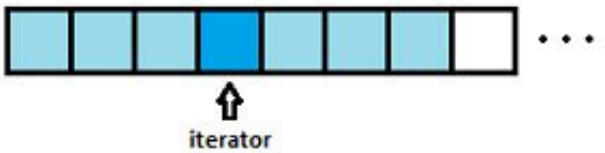
Implementace na poli (cyklicky): Máme pole typu stejného jako data a indexy First a Last. Indexy neukazují přímo do pole, nýbrž je proveden přepočet (Index mod Delka pole) abychom využili celé pole. Tato fronta má omezenou délku.

- **Konstruktor:** Alokace pole a indexů, nastavení Last na -1 a First na -1
- **Modifikátor:** Pokud ((Last + 1) mod Delka) != (First mod Delka), Push provede Last++ a zapíše data do pole na tento index, jinak chyba (plná fronta). Pokud First = -1, pak First = 0 (první prvek).
- **Selektor:** Front vrátí prvek pole pod indexem First, pokud First <= Last. Pop jako Front, ale před návratem provede First++.

Speciálním případem je například prioritní fronta, která řadí prvky ve frontě podle priority

Seznam

Interface - Iterator nad seznamem: Add, Remove, Get; First, Next, IsLast, případně další (Previous, Last, IndexOf, ...)



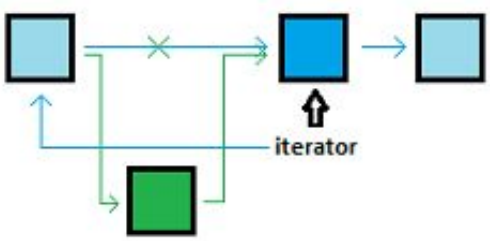
- **Add** vkládá data na pozici iterátoru. Složitost dle implementace.
- **Remove** maže data na pozici iterátoru

- **Get** vrátí data na pozici iteratoru
- **First** vrátí iterátor na začátek seznamu
- **Next** posune iterátor dále
- **IsLast** indikuje koncová prvek
- (Previous v obousměrných seznamech posune iterator na předchozí prvek)
- (Last posune iterator na poslední prvek)
- (IndexOf vyhledá prvek ve struktuře)

= struktura, do které je možné libovolně zapisovat a číst (na libovolné pozici)

Iterator: ukazatel do struktury

Implementace objektově - nástřel:



Prvky ukazují na následující prvky seznamu. Iterátor je dobré implementovat tak, že ve skutečnosti ukazuje na předchozí prvek (Pokud tedy seznam není obousměrný, pak máme metodu Previous a je to jedno). Metoda add vloží prvek za prvek, na který ukazuje iterátor tak, že pouze upraví ukazatele prvků:

Remove je opačný proces, Get vrátí data (Tedy Iterator.Next.Data), První prvek seznamu si uložíme, abychom se mohli rychle vracet, stejně jako poslední prvek. Next provede Iterator.Next = Iterator.Next.Next. Kontrolujeme, zda nejsme mimo seznam.

- Obousměrný seznam: Prvky mají i ukazatele na předchozí prvky, a je tedy možné se po nich pohybovat na obě strany.
- Kruhový seznam: Poslední prvek neukazuje na null, ale na první prvek. Tyto seznamy jsou použitelné ve speciálních případech (například nastavení segmentů sedmisegmentovky pro digitální hodiny)

Implementace na poli (nástřel): Pole má nevýhodu, že je nutné přesunout prvky doprava od iterétoru, pokud vkládáme prvky. Při mazání lze prvky posunout doleva, nebo do pole uložit hodnotu, která se zcela jistě nebude v poli nebude vyskytovat, nebo vytvořit stínové pole, ve kterém si smazané prvky označíme, a při posunu iterátoru je poté vynecháme. Tento přístup je dobrý pokud pole často upravujeme.
