ADATSZERKEZETEK ÉS ALGORITMUSOK

LIFO, FIFO Láncolt megvalósítás
„LIFO, FIFO, Szekvenciális adatszerkezetek”

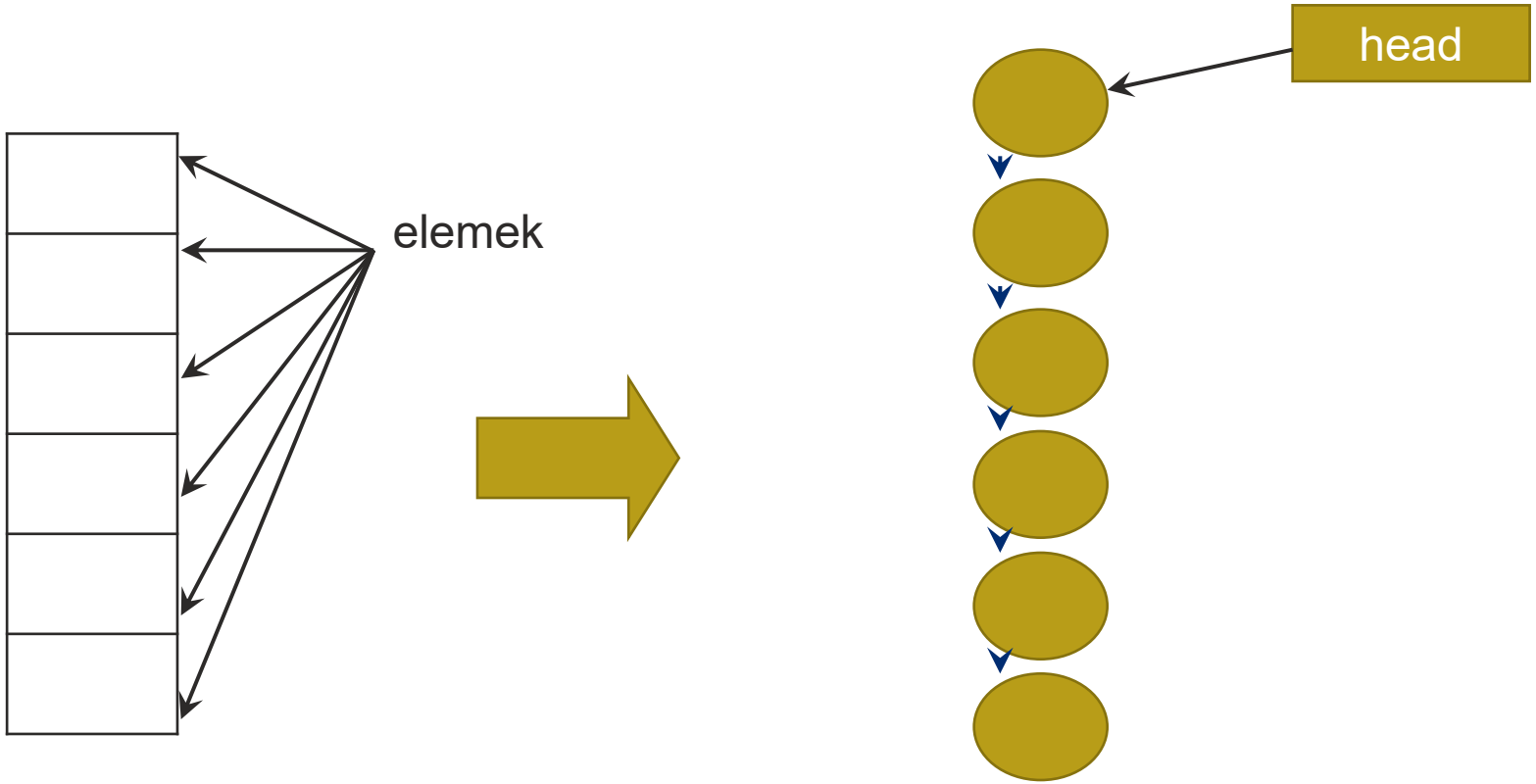
Verem – Láncolt ábrázolás

- A fix méretű megvalósítás korlátai:
  - Ha betelik a verem, nem tudjuk további elemek tárolására használni.
  - Ha túl nagy vermet hozunk létre, feleslegesen foglaljuk a memóriát.
- A megoldás természetesen a változtatható méretű (dinamikus) ábrázolás.
  - A dinamikus megvalósítást láncolással oldjuk meg.
  - Ez az jelenti, hogy létrehozunk egy osztályt a veremhez, amely tartalmazza az értéket, és egy pointerrel mutat a következő elemre.

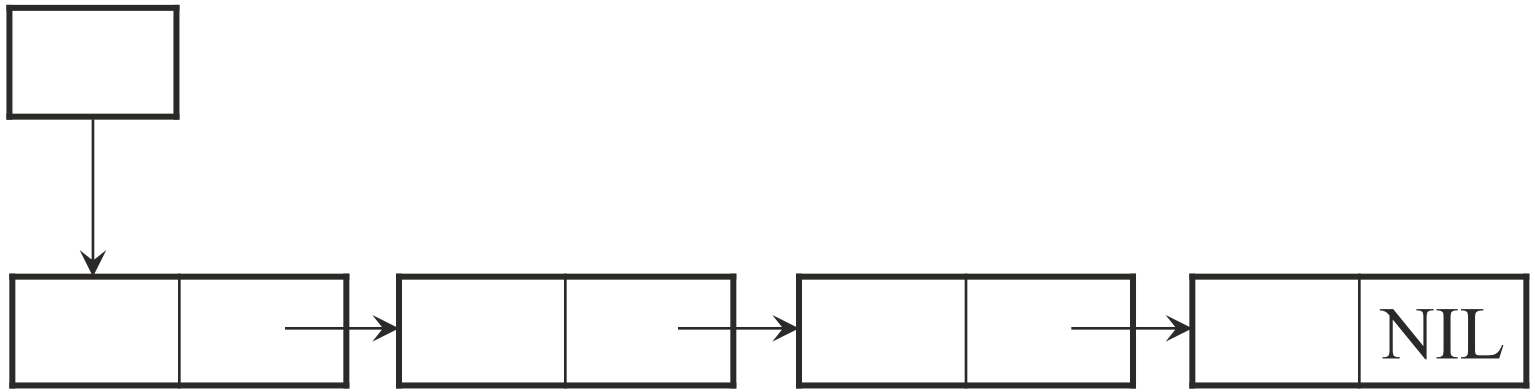
```
class Node{
  public:
    int value;
    Node* pNext;
};
```

Verem – szerkezete

- Tömb alapú megvalósítás → Láncolt ábrázolás



Verem – Láncolt ábrázolás



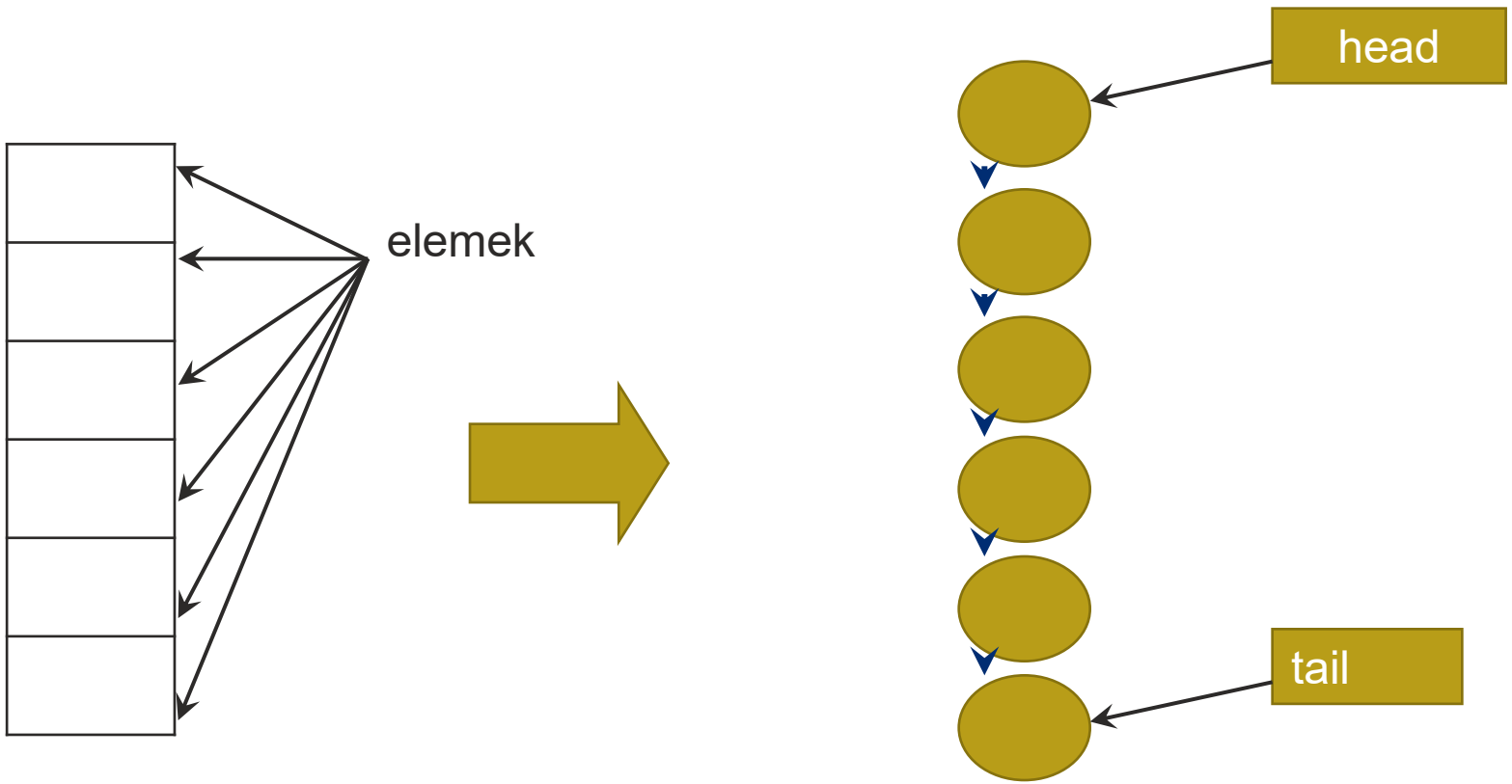
Sor – változó méretű megvalósítás

- A fix méretű megvalósítás korlátai:
  - Ha betelik a sor, nem tudjuk további elemek tárolására használni.
  - Ha túl nagy sort hozunk létre, feleslegesen foglaljuk a memóriát.
- Egy lehetséges megoldás a láncolásos implementáció.
  - Létrehozunk `Node` típusú elemeket, amely tartalmazzák az értéket, és egy pointerrel mutatnak a rákövetkező elemekre.

```
struct Node {
 int value;
 Node *pNext;
};
```

Sor megvalósítás

- A tömb alapú megvalósítás → Láncolt ábrázolás



Sor – változó méretű megvalósítás

- A `head` és `tail` változók ebben az esetben nem indexek lesznek, hanem mutatók, mégpedig az első ill. az utolsó elemre mutatók. (`Node*`)