# ADATSZERKEZETEK ÉS ALGORITMUSOK

Szekvenciális adatszerkezetek
„LIFO, FIFO, Szekvenciális adatszerkezetek”

# Szekvenciális adatszerkezetek

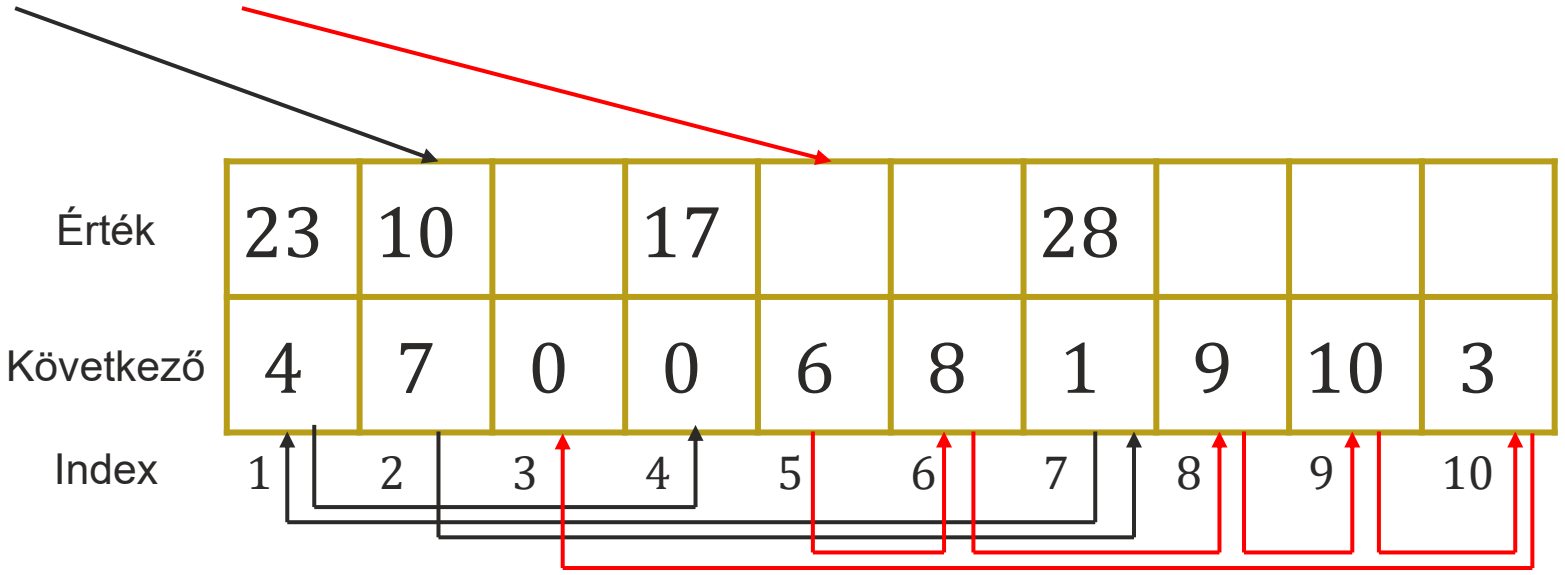
- Szekvenciális adatszerkezetben az egyes adatelemek egymás után helyezkednek el
  - Van egy logikai sorrendjük
- Az adatok között egy-egy jellegű a kapcsolat
  - Minden adatelem csak egy helyről érhető el és az adott elemtől csak egy másik látható
- Két kitüntetett elem
  - az első
  - az utolsó

# Szekvenciális adatszerkezetek

- Ez egy homogén adatszerkezet, azaz azonos típusú véges adatelemek sorozata
  - Jelölése : $L = (a_1, a_2, \ldots\ a_n\ )$
  - Ha $n = 0$, akkor $L = ()$ az üres lista.
- A láncolt lista olyan adatszerkezet, amelynek minden eleme tartalmaz egy (vagy több) mutatót (hivatkozást) egy másik, ugyanolyan típusú adatelemre
  - Ez a „következő” elem logikailag
- A lánc első elemének a címét a lista feje tartalmazza
  A listafej nem tartalmaz információs részt
- A lánc végét az jelzi, hogy az utolsó elemben a rákövetkező elem mutatója üres

# Reprezentációs szint

- Fix kapacitású ábrázolás
  - tömbben, a logikai sorrendet indexek mutatják, a szabad helyek is listában:
- L: 2 SZH: 5



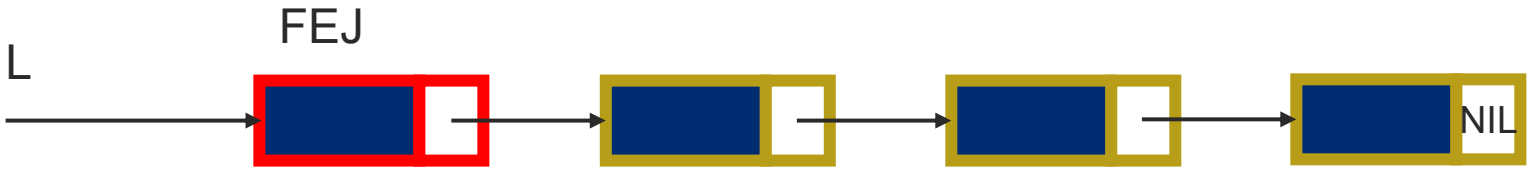
# Dinamikus, láncolt ábrázolás

- Az adatok száma nem ismert előre
  - Nem tudunk, vagy nem akarunk feleslegesen helyet foglalni az adatoknak
  - A feladat dinamikusan változik

# Láncolt ábrázolás

- Egyirányú láncolt lista
  - Fejelem nélkül

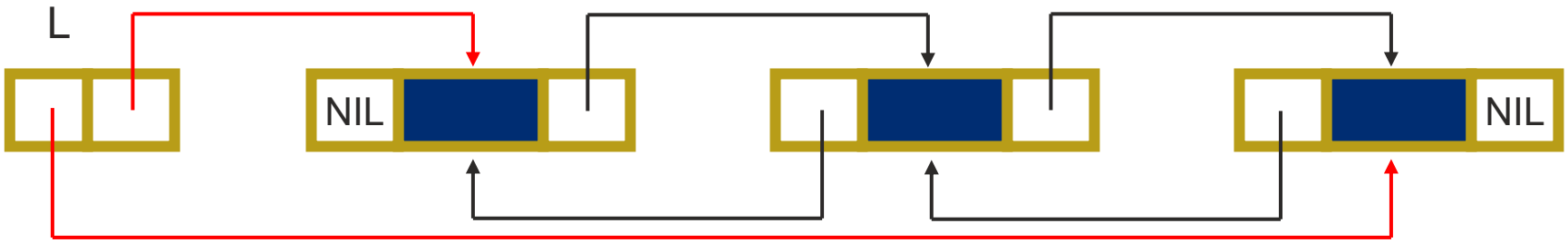

  - Fejelemmel: fejelem mindig létezik, ha üres a lista, akkor is




- Mindig van egy aktuális elemre mutató is, ez része a megvalósításnak

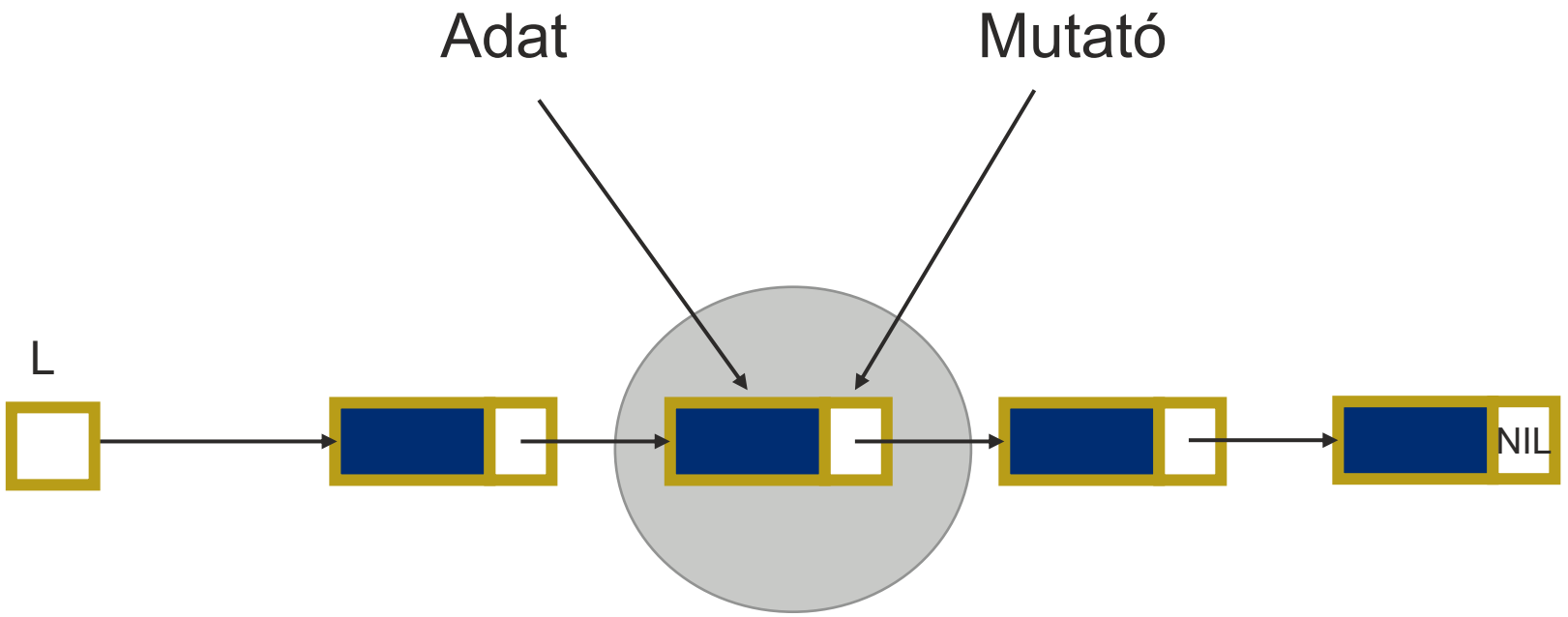
# Láncolt ábrázolás

- Kétirányú láncolt lista



# Láncolt ábrázolás

- A lista eleme

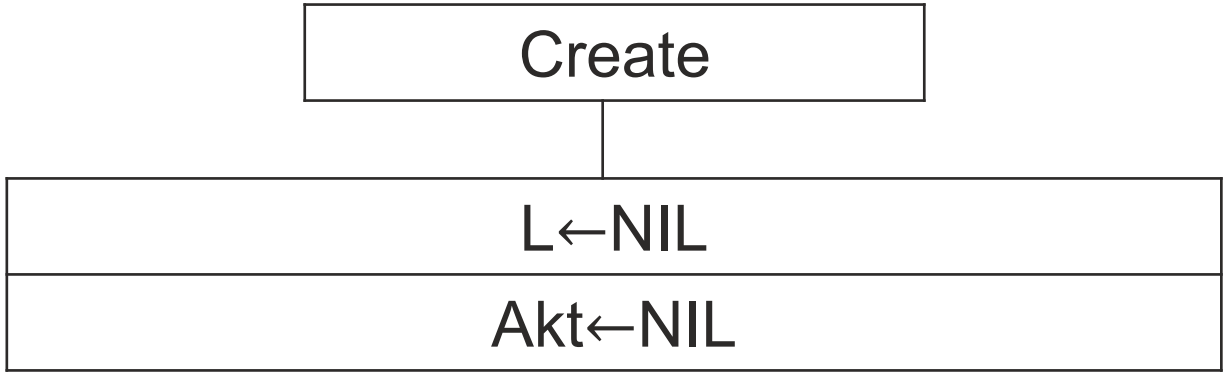


# Egyszerű lista – műveletek

- A Lista típus komponensei:
  - L
    - A lista első elemének mutatója,
  - Akt
    - A lista aktuális elemének mutatója

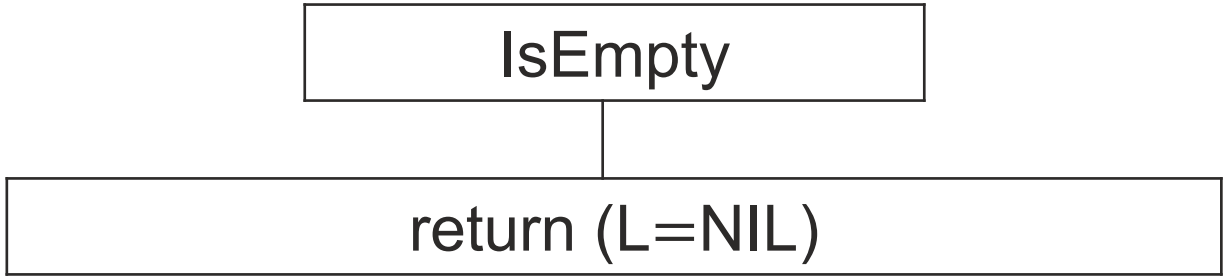
# Létrehozás

- Üres listát ad vissza

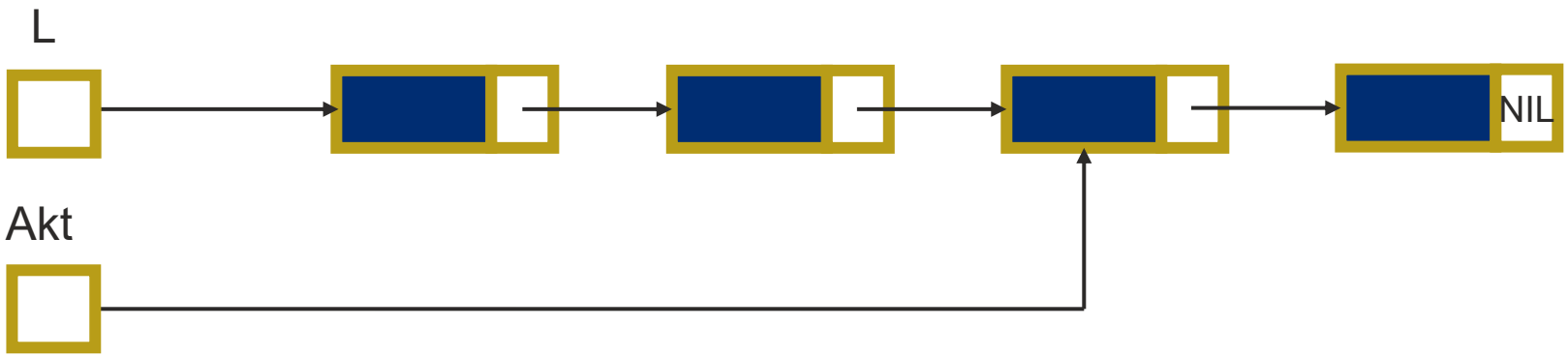


# Üres lista lekérdezése

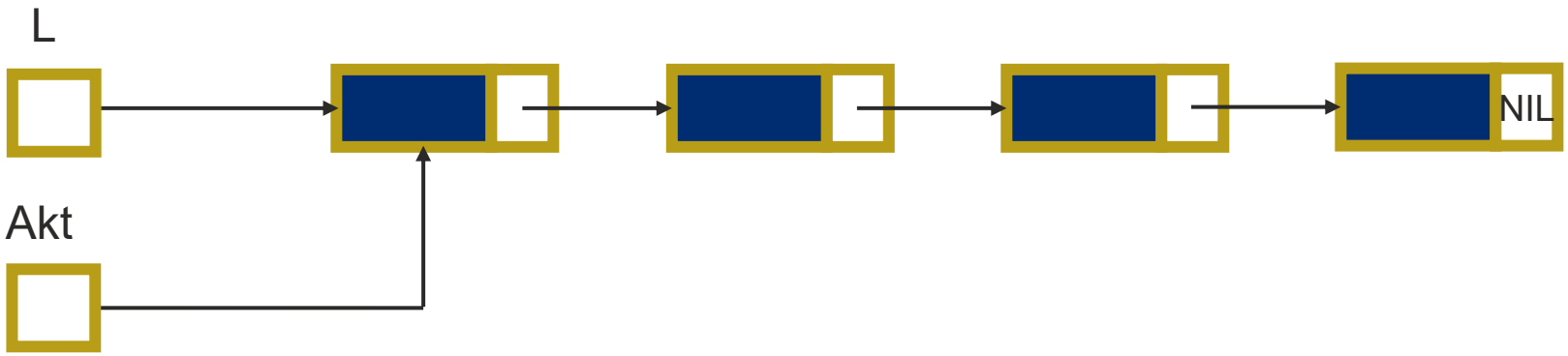
- Logikai értéket ad vissza



# Első elemre áll

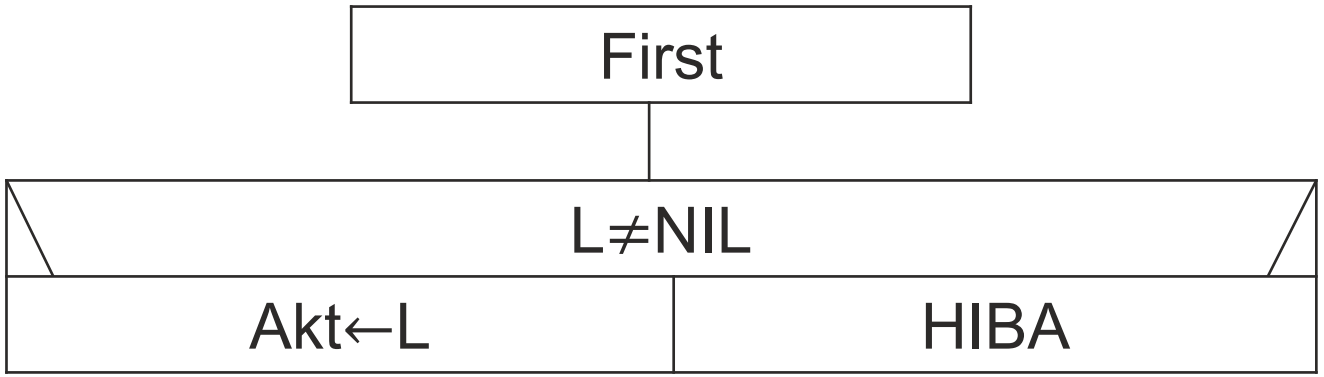


# Első elemre áll

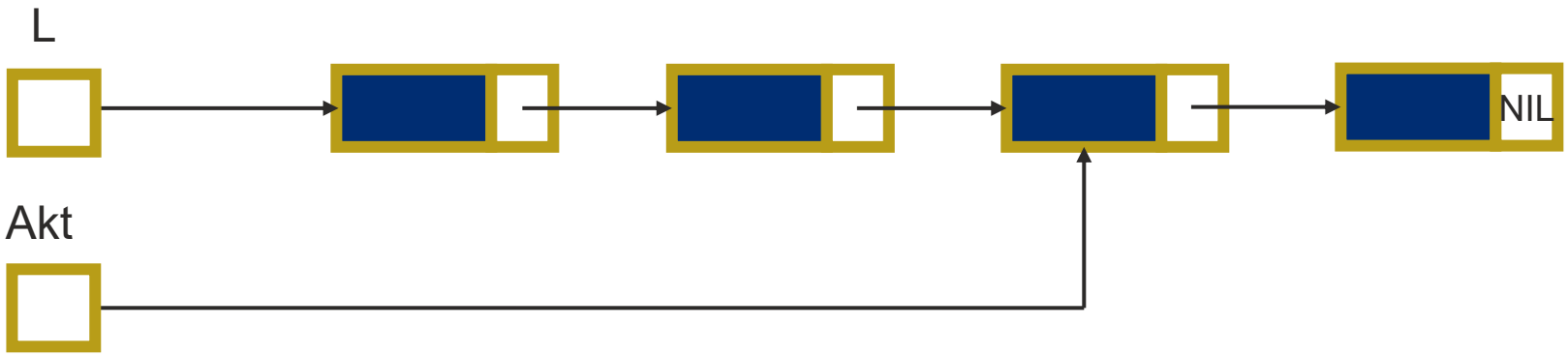


# Üres lista lekérdezése

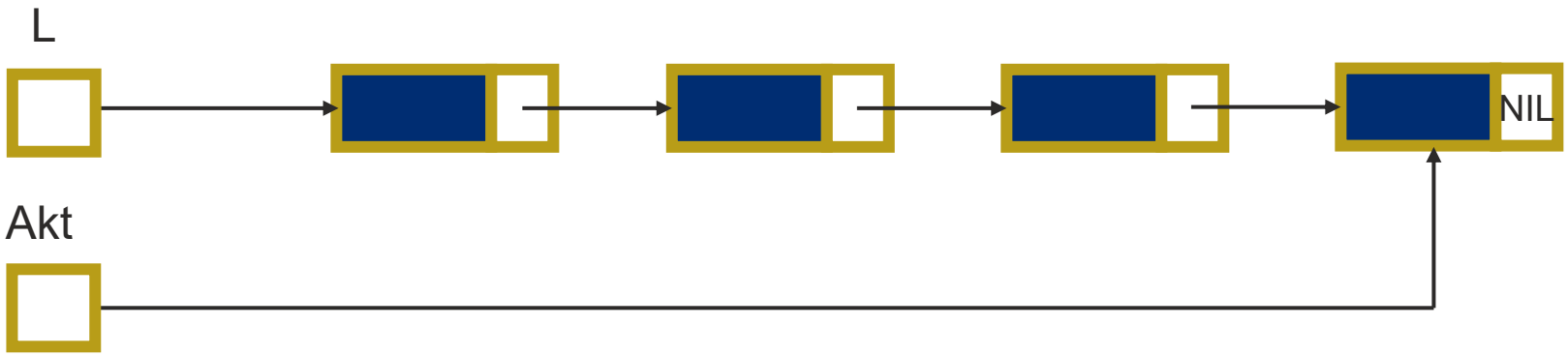
- Üres lista esetén hiba



# Következő elemre áll

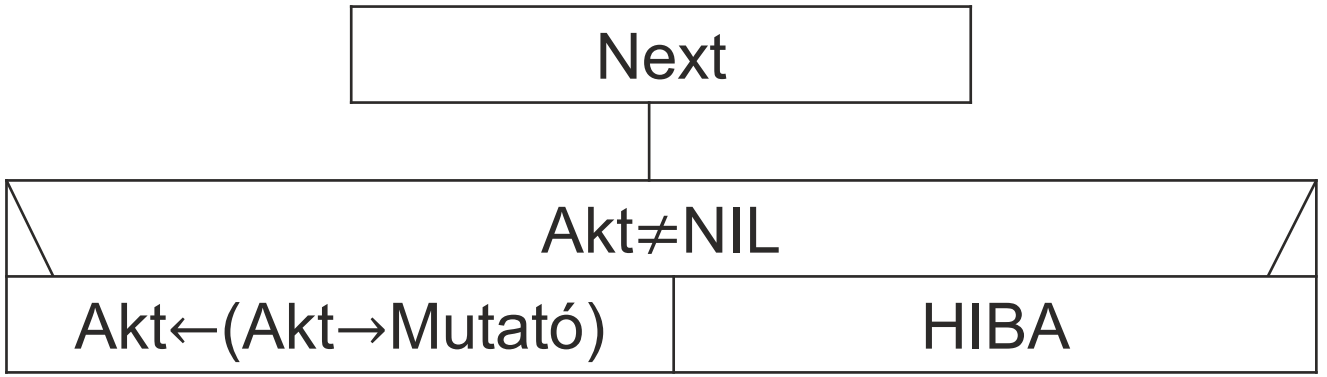


# Következő elemre áll



# Következő elemre áll

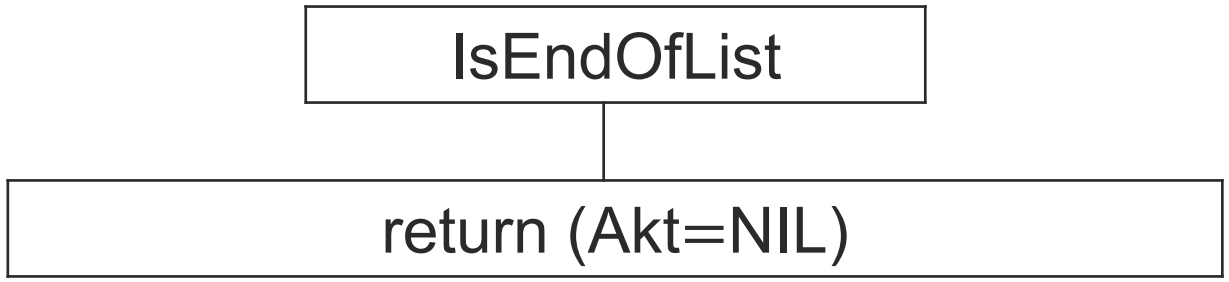
- A lista utolsó elemére kiadott Next hatása Akt=NIL lesz




Adat Mutató

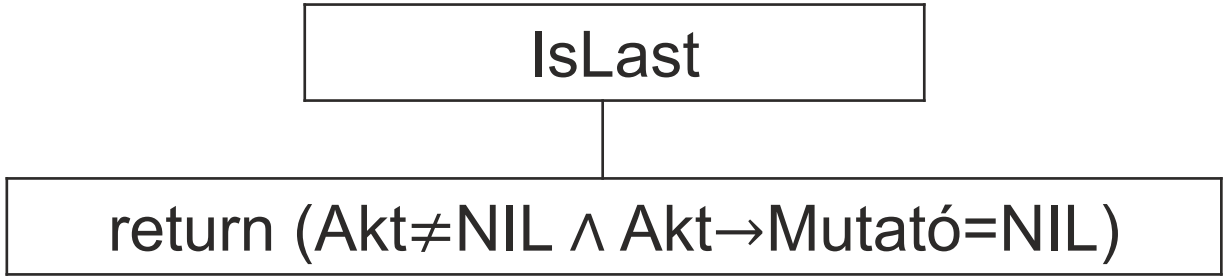
# Lista végének lekérdezése

- Logikai értéket ad vissza



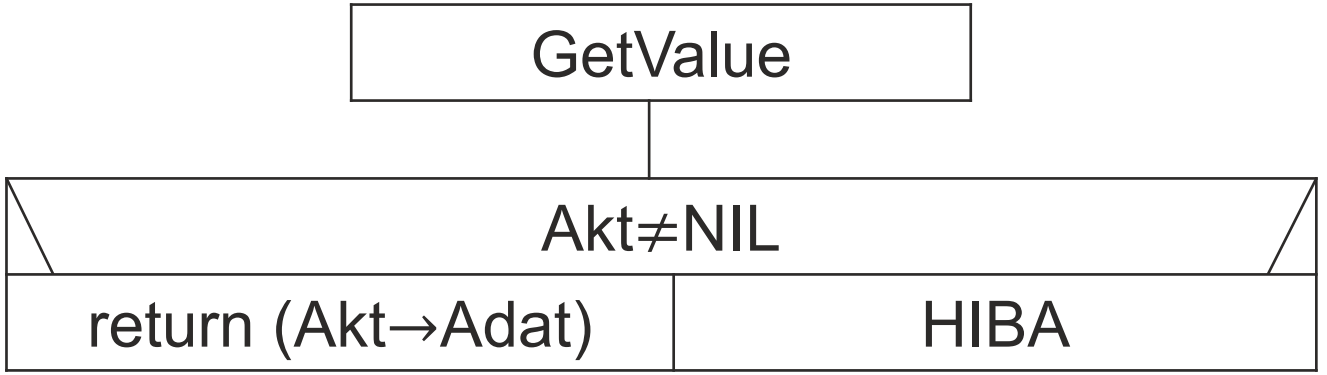
# Az aktuális az utolsó elem-e

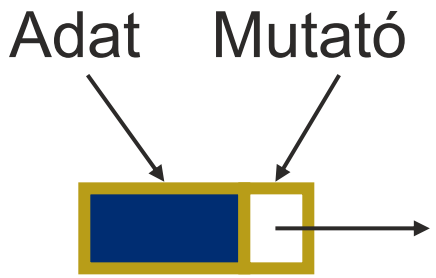
- Logikai értéket ad vissza



# Aktuális elem értéke

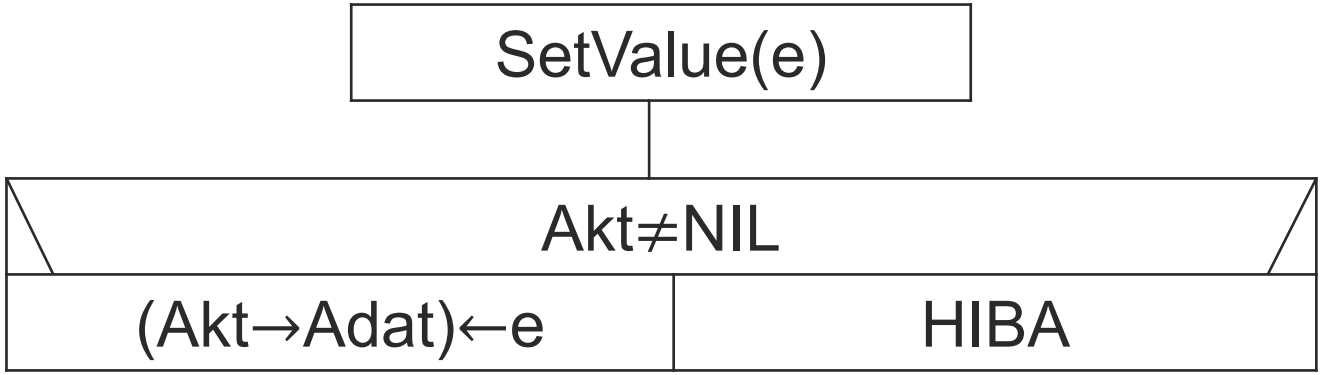
- Az aktuális elem értékével tér vissza



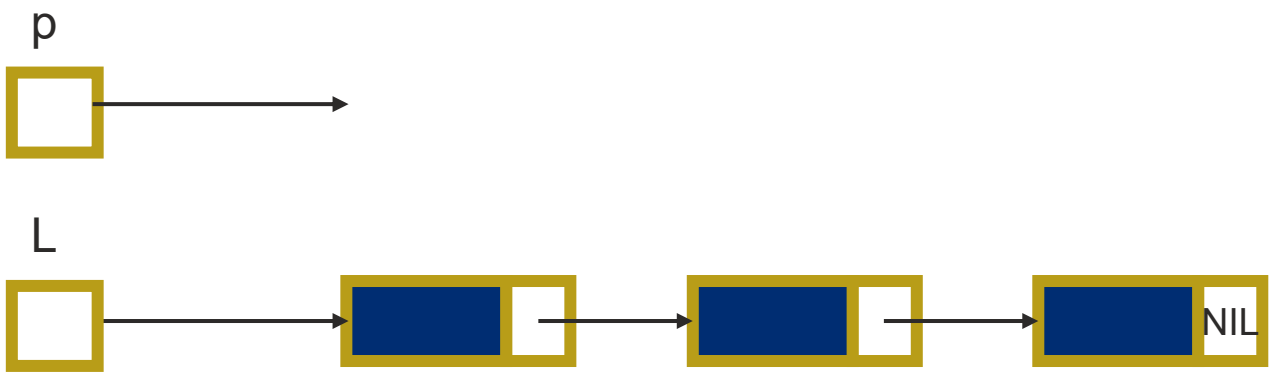
# Aktuális elem módosítása

- Az aktuális elem megváltoztatása e -re



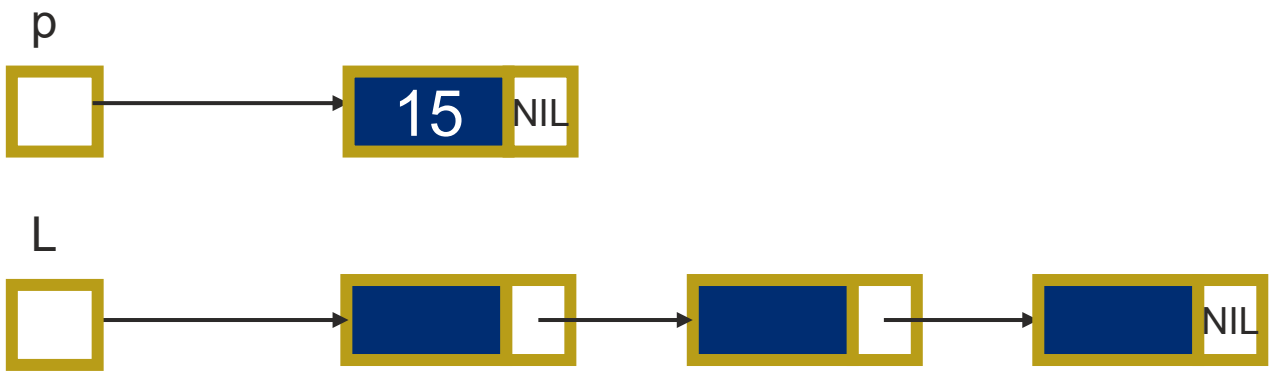
# Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal



# Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása

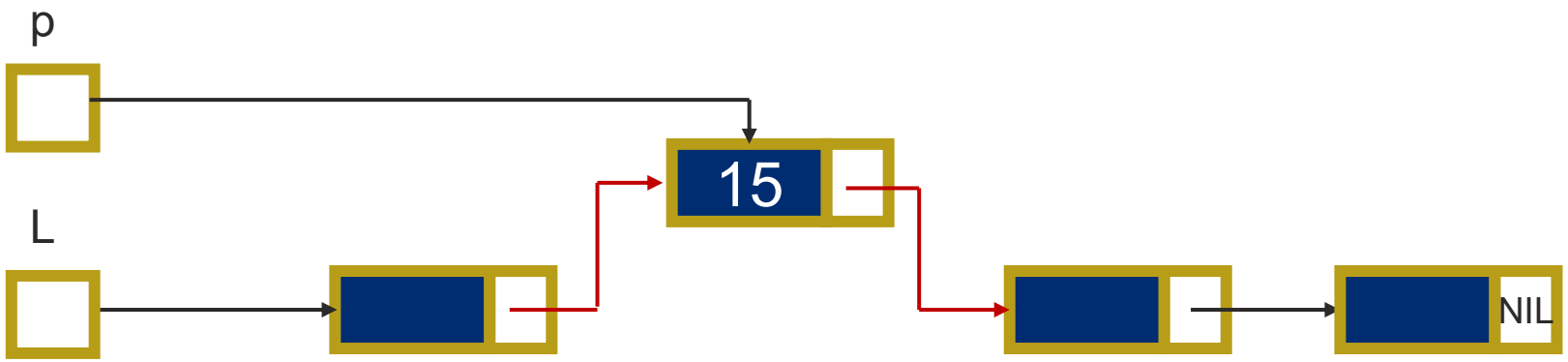


# Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása
  - Új elem befűzése a láncolatba

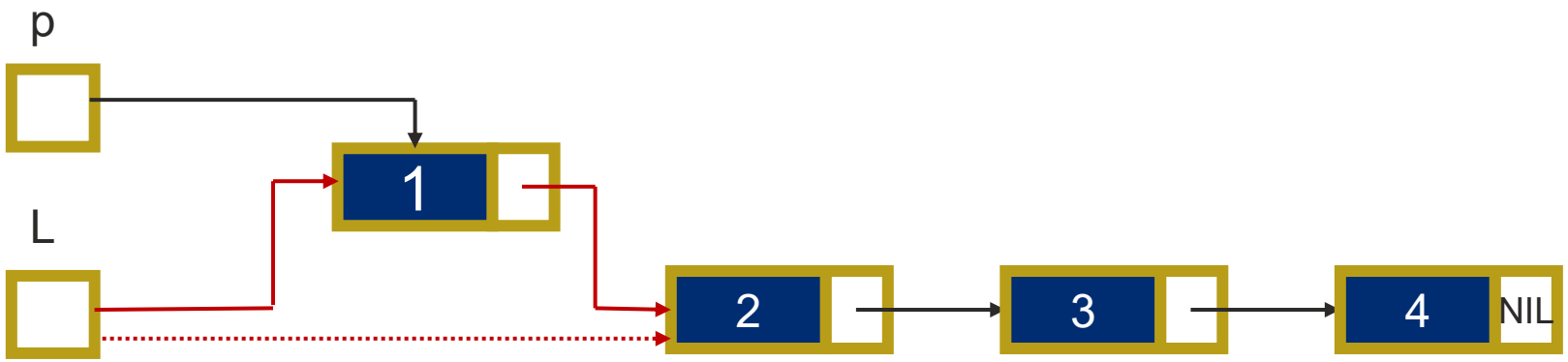
# Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása
  - Új elem befűzése a láncolatba



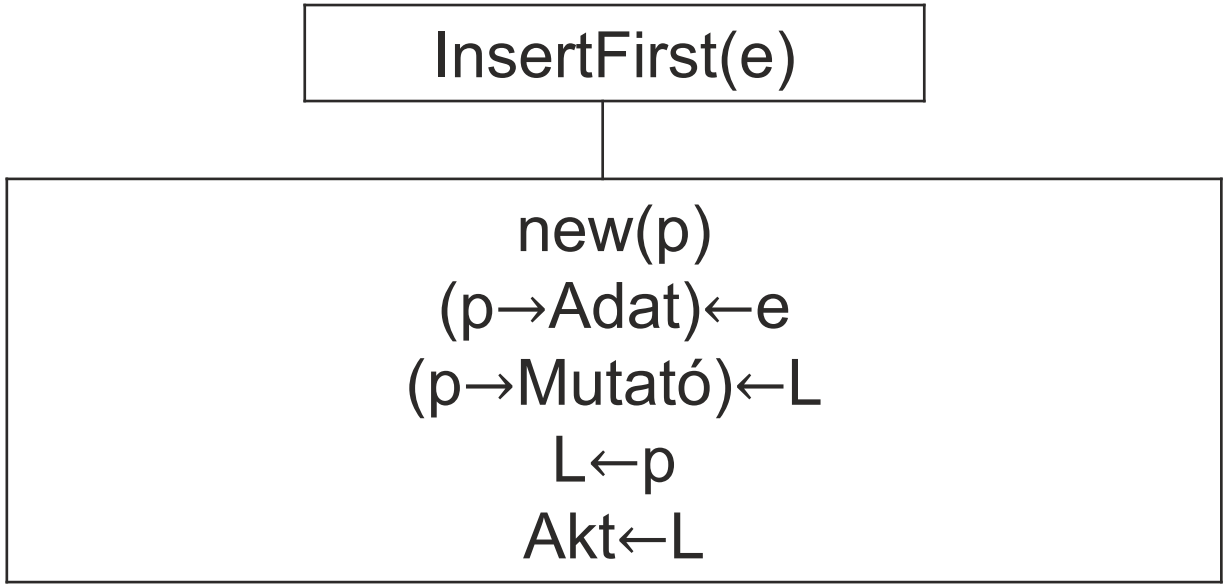
# Listaelem beszúrása

- Beszúrás első elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az első



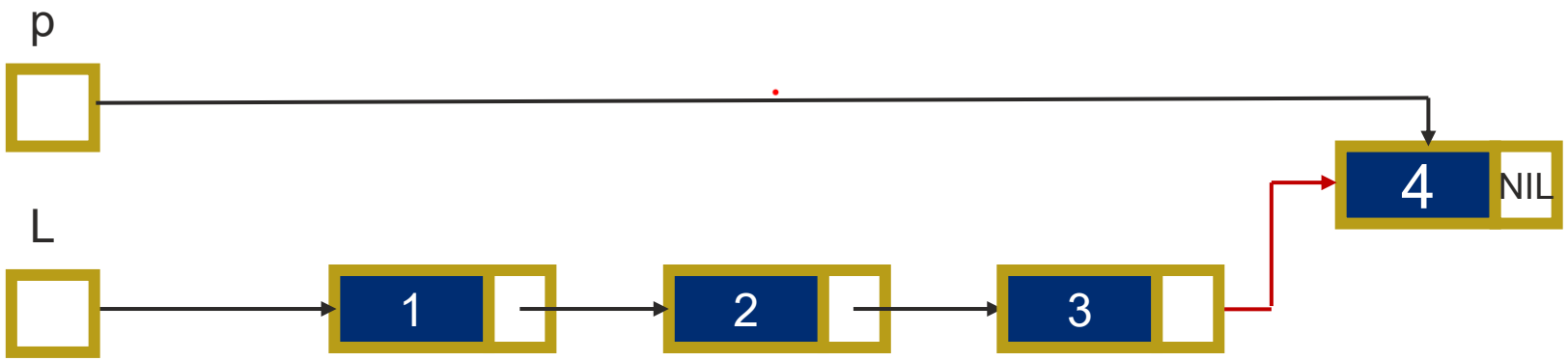
# Listaelem beszúrása

- „e” adatelem beszúrása első elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az első



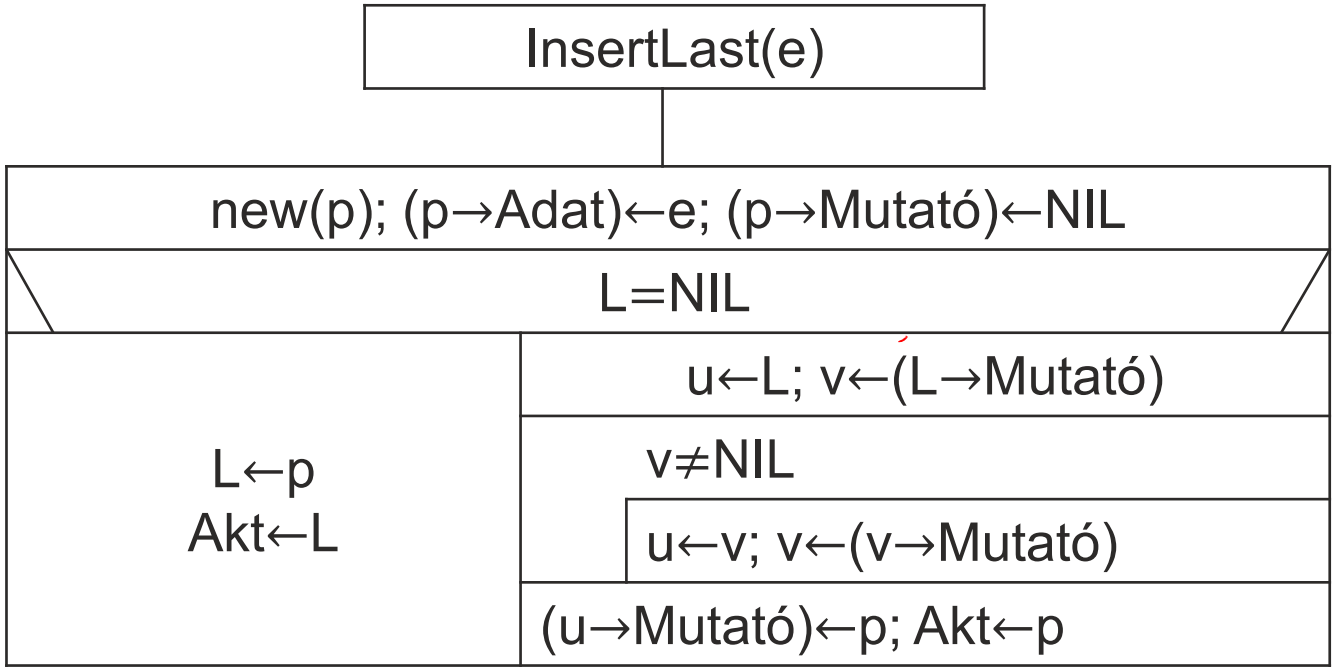
# Listaelem beszúrása

- Beszúrás utolsó elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az utolsó



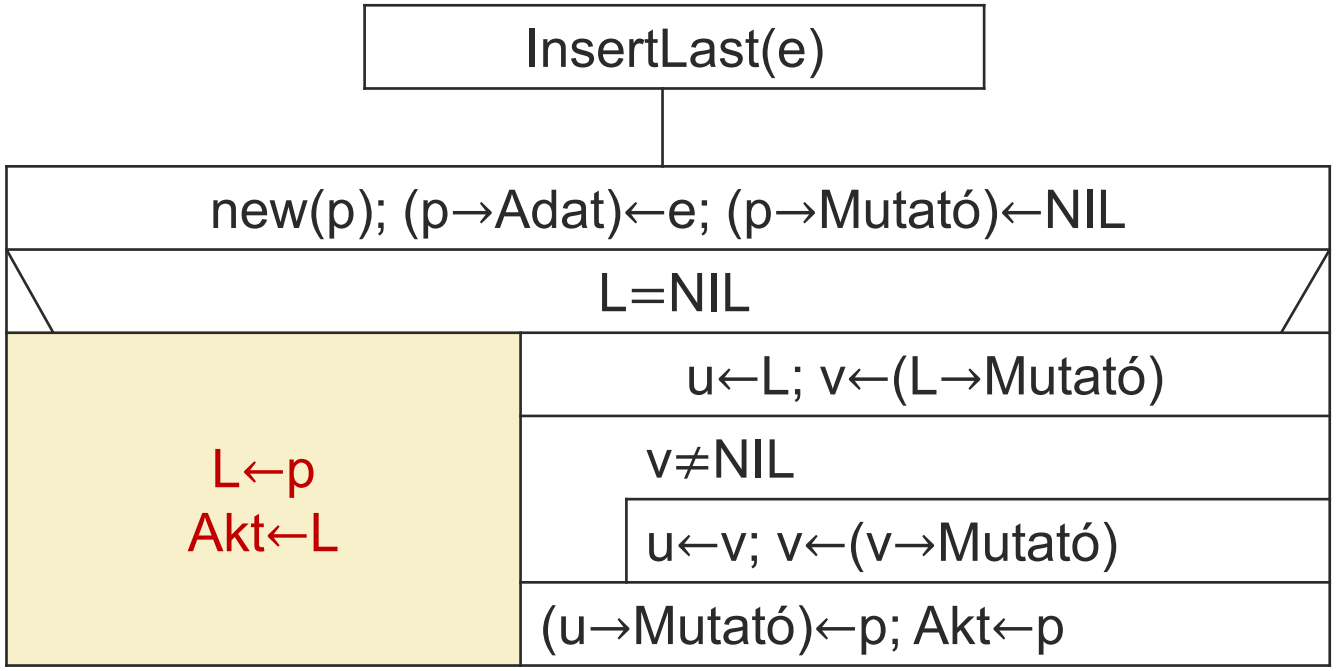
# Listaelem beszúrása

- „e” adatelem beszúrása utolsó elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az utolsó



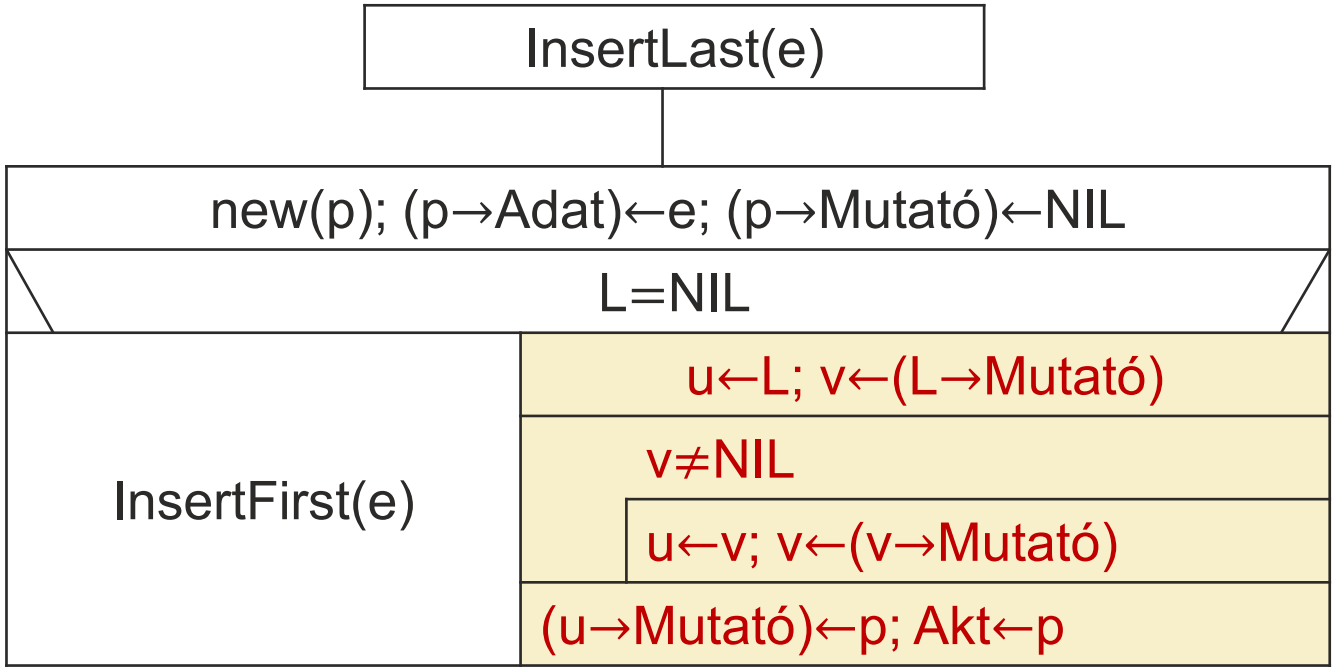
# Listaelem beszúrása

- Ha a lista üres
  - Ami egyezik az InsertFirst(e) algoritmusával
    - Azonban itt nem lehet behelyettesíteni, mert az új csomópont már létrejött előbb



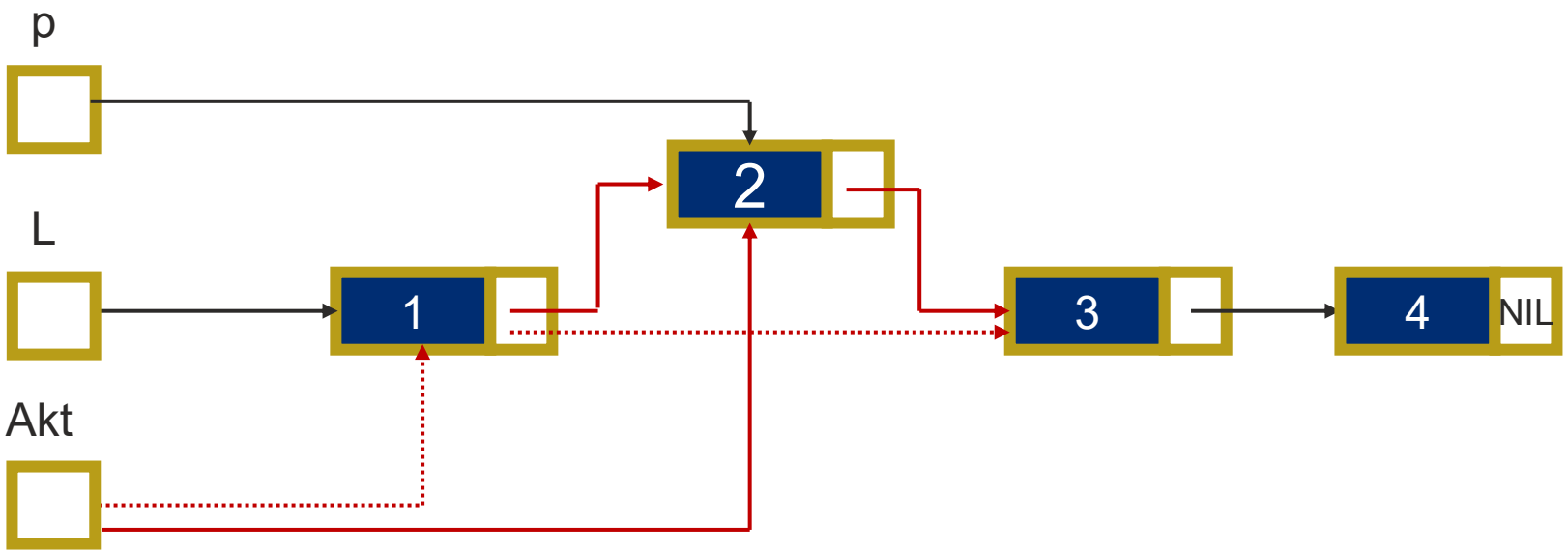
# Listaelem beszúrása

- Ha a lista nem üres
  - Meg kell keresni az eredeti lista utolsó elemét
  - Amögé kell beszúrni az új elemet



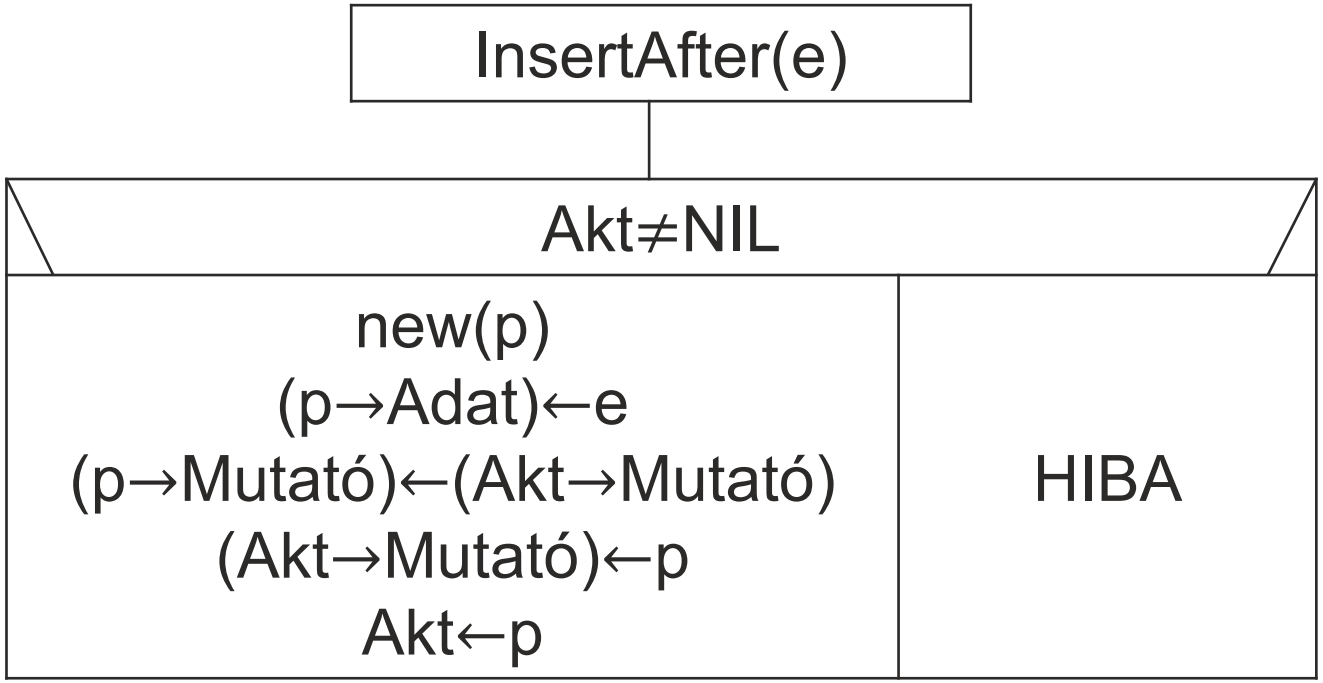
# Listaelem beszúrása

- Beszúrás az aktuális elem után



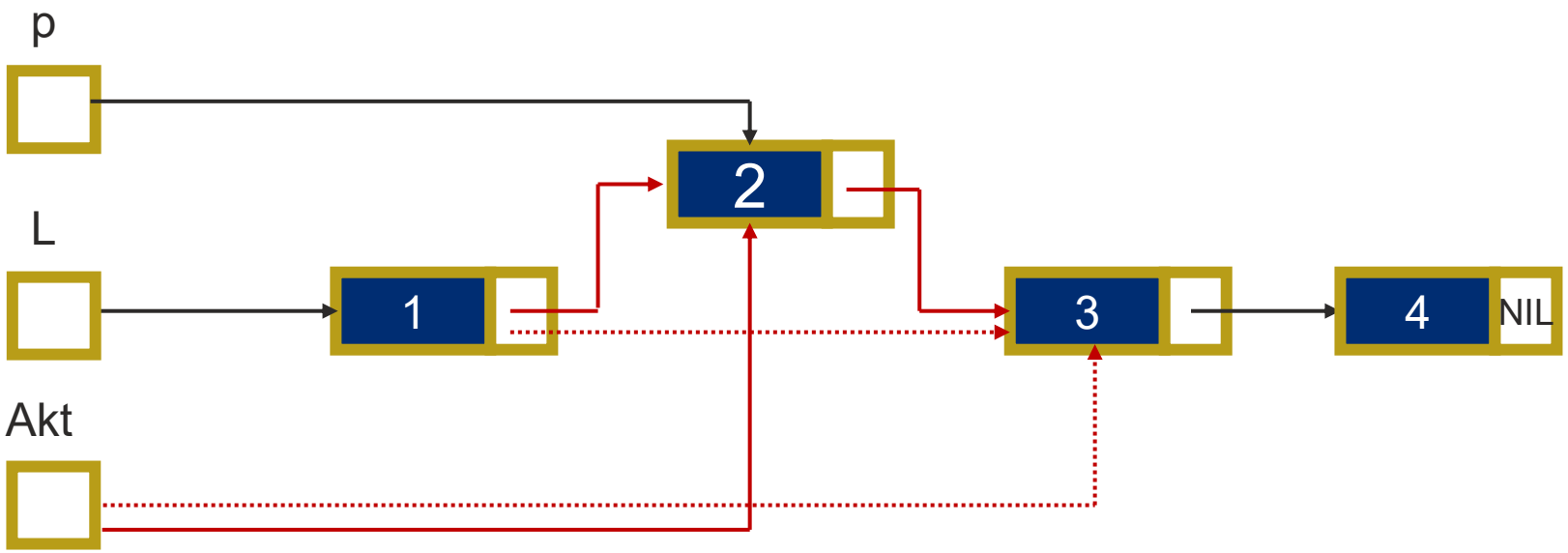
# Listaelem beszúrása

- Beszúrás az aktuális elem után
  - Ha nincsen aktuális elem, az hiba
  - Az újonnan beszúrt lesz az aktuális elem



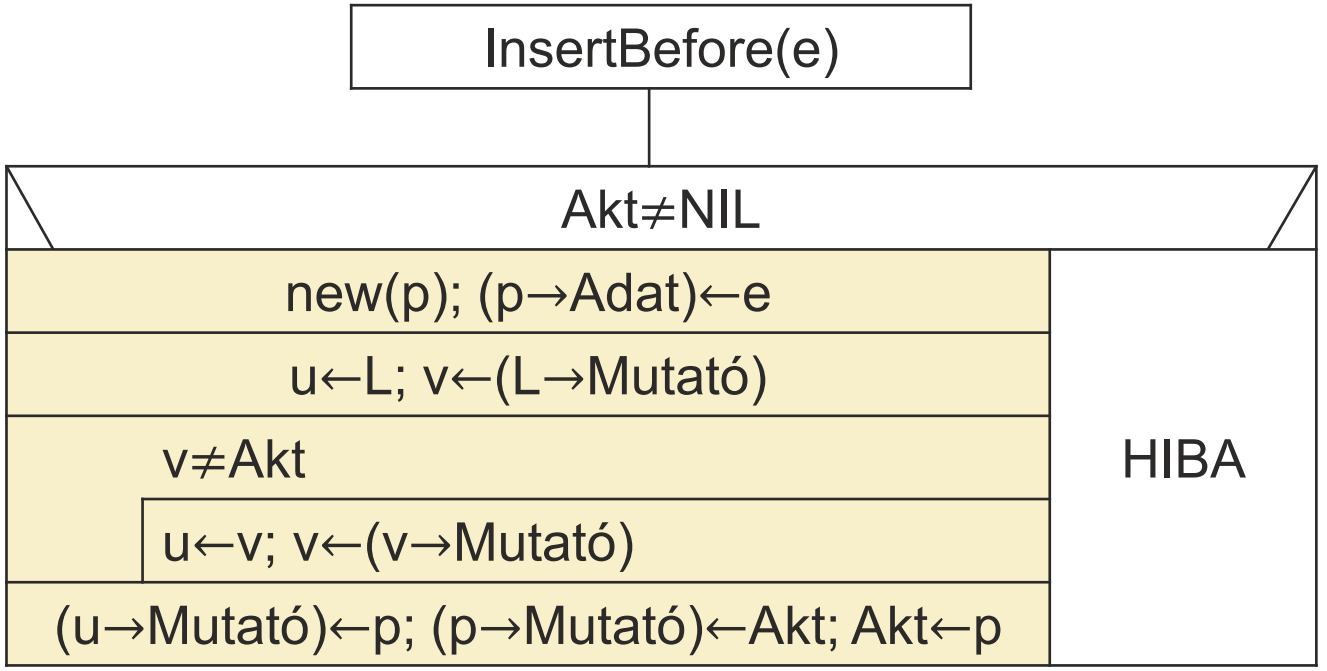
# Listaelem beszúrása

- Beszúrás az aktuális elem elé



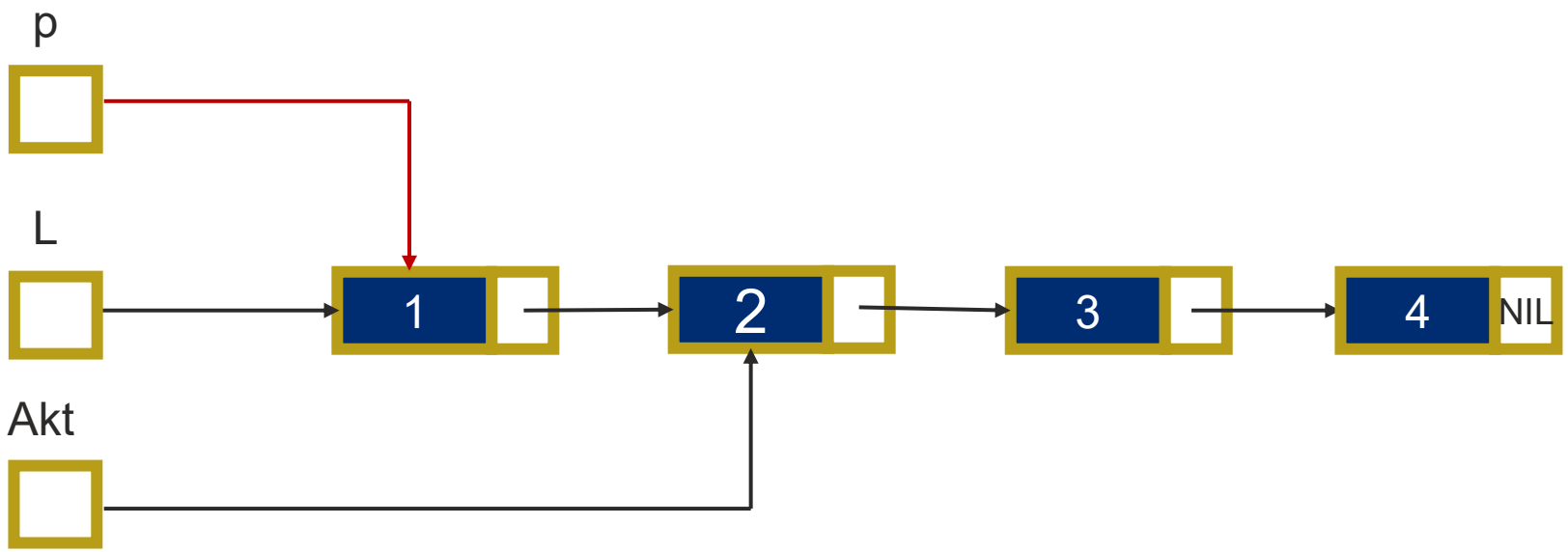
# Listaelem beszúrása

- Beszúrás az aktuális elem elé
  - Ha nincsen aktuális elem, az hiba
  - Az újonnan beszúrt lesz az aktuális elem



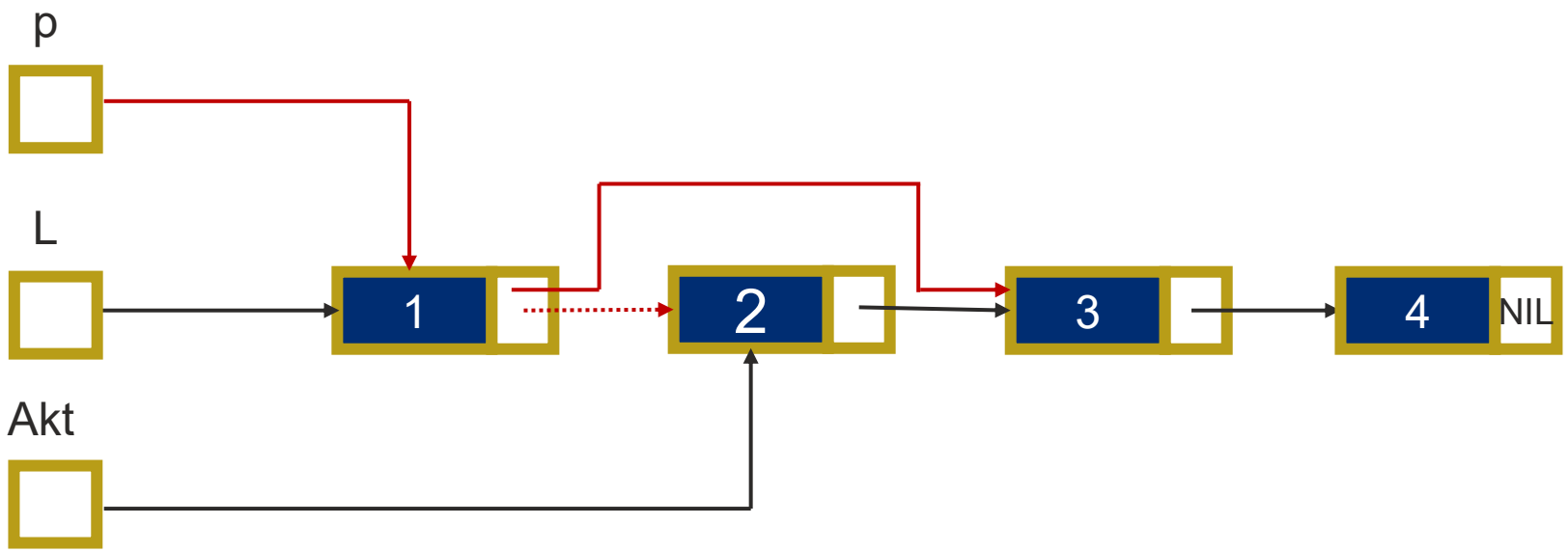
# Listaelem törlése

- Aktuális elem törlése
  - Törlendő elem megelőzőjének megkeresése
  - Láncolás megváltoztatása (átláncolás)



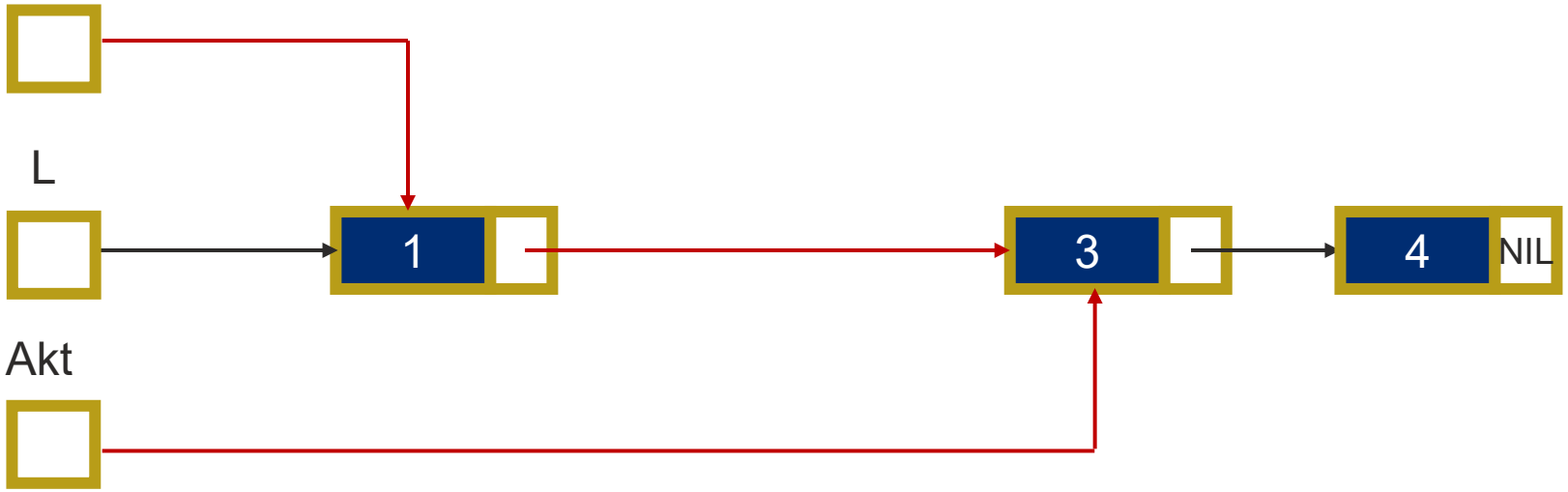
# Listaelem törlése

- Aktuális elem törlése
  - Törlendő elem megelőzőjének megkeresése
  - Láncolás megváltoztatása (átláncolás)



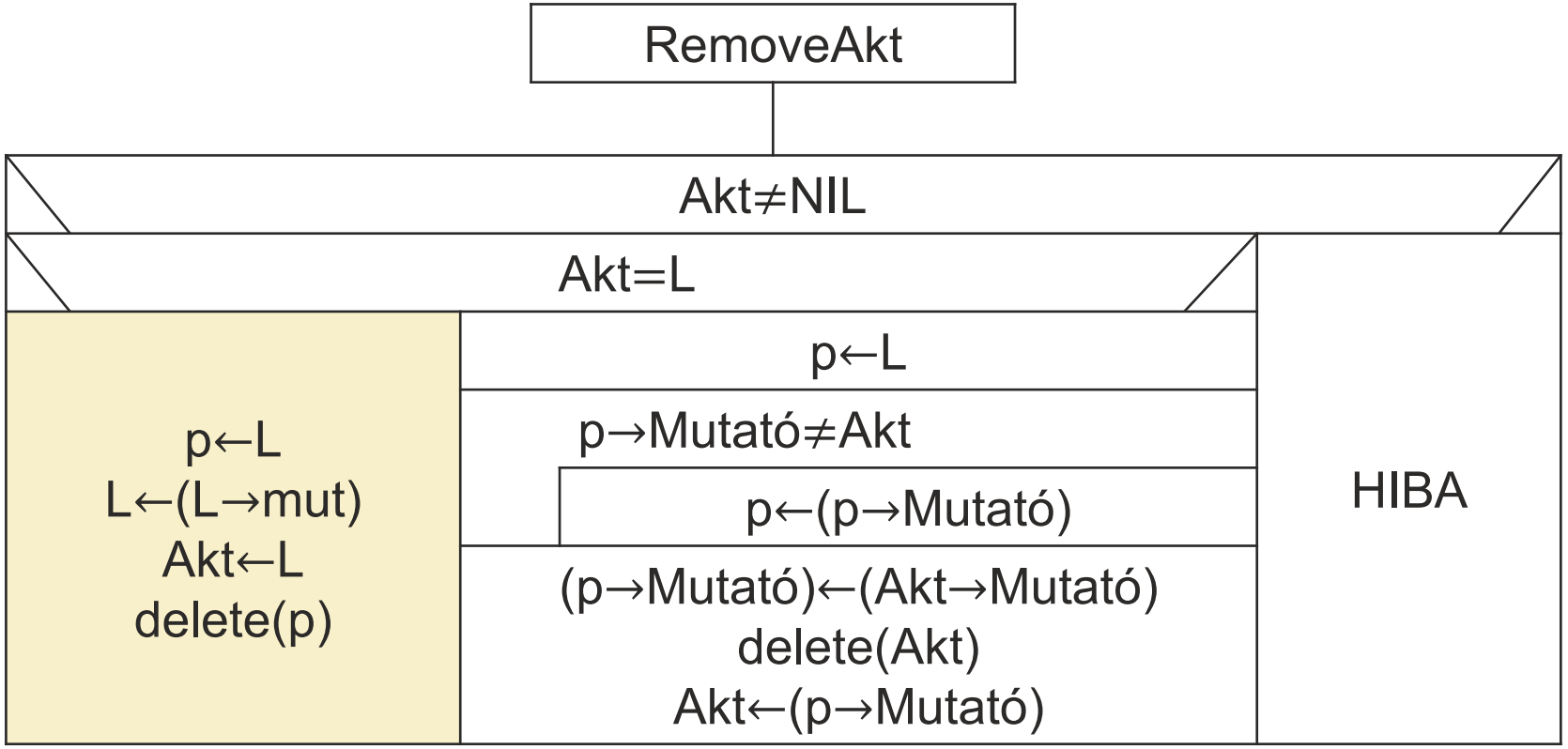
# Listaelem törlése

- Aktuális elem törlése
  - Törlendő elem megelőzőjének megkeresése
  - Láncolás megváltoztatása (átláncolás)
  - Memóriából eltávolítás (törlés)
  - Akt beállítása



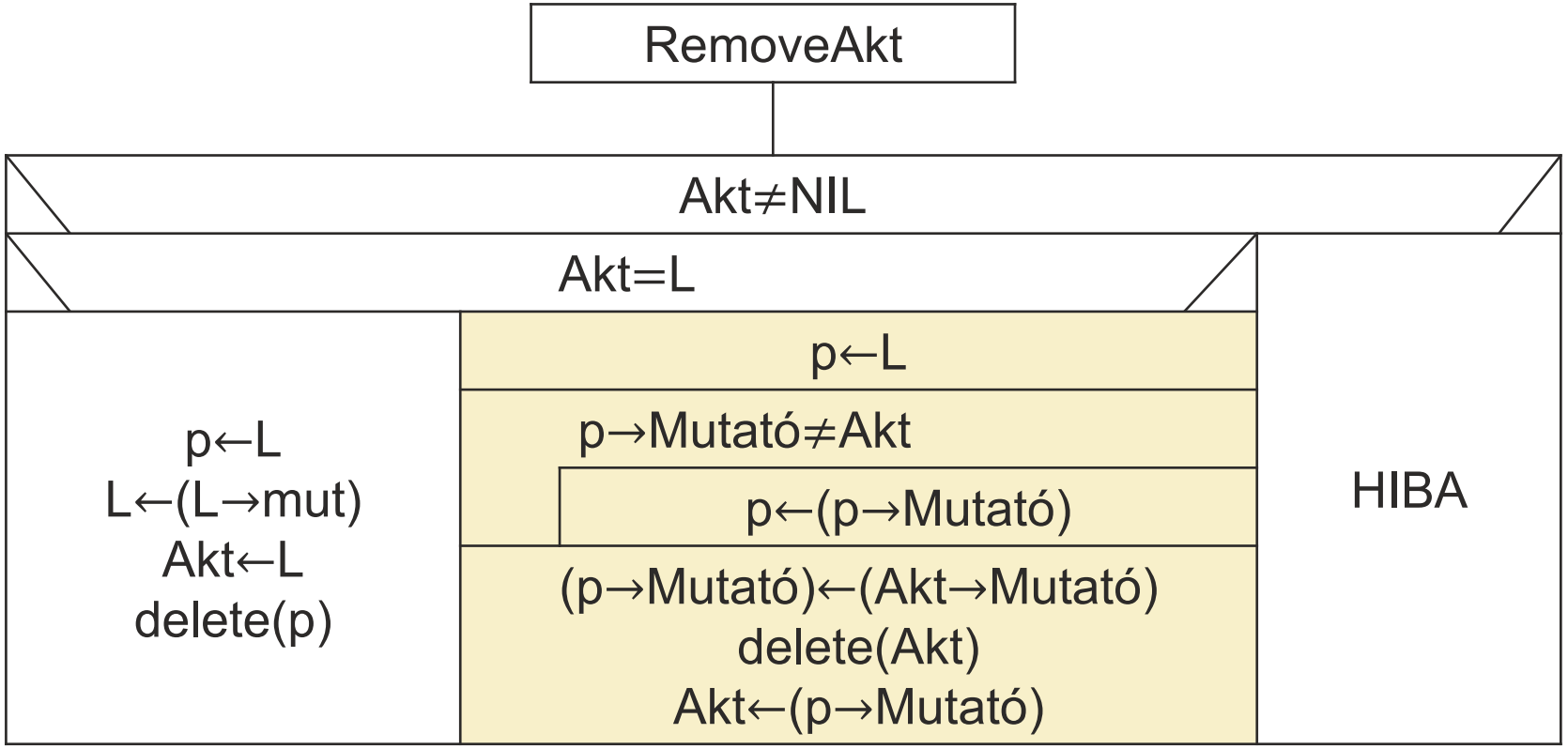
# Listaelem törlése

- Aktuális elem törlése
  - Ha az aktuális az első



# Listaelem törlése

- Aktuális elem törlése
  - Ha az aktuális nem az első



# Egyszerű lista – Műveletek

- Megjegyzések – lehetőségek
  - Az Akt nem változik a módosítás során
  - További műveletekre példa
    - Teljes lista törlése
    - Listák összefűzése,
    - Elemszám lekérdezése
  - Ebben az implementációban nem hatékony a megvalósítás
    - Last, Remove, InsertBefore, InsertLast
  - Hatékonnyá tehető
    - Kétirányú láncolással