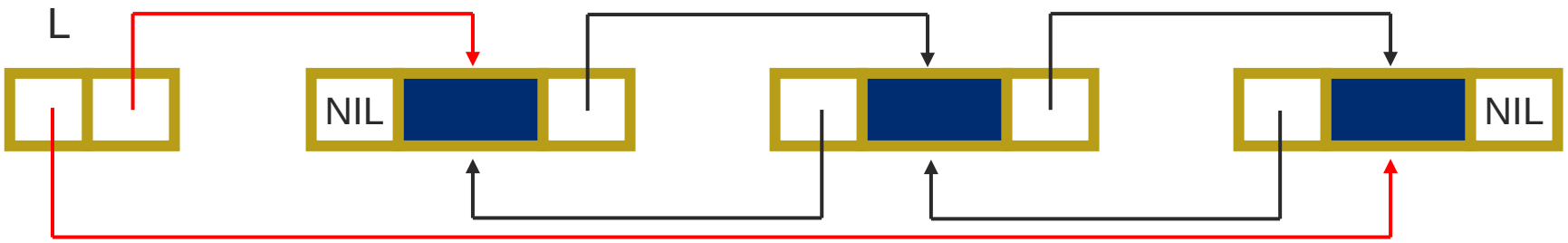
ADATSZERKEZETEK ÉS ALGORITMUSOK

Láncolt Lista Gyakorlati megvalósítás

Láncolt listák

- **Egyirányú láncolt lista**
  pl.: A láncolt dinamikus sor nagyon hasonló egy ilyen listához
  (Csak a műveletek mások.)

- **Kétirányú láncolt lista**
  (Most ezzel fogunk foglalkozni)



Kétirányú láncolt lista osztály

```
class List {
   private:  // Ez a default láthatóság, de szebb, ha kiírjuk
      struct Node {
         …
      };
   public:
         …
};
```

- A `Node` típus a listának egy belső típusa (enkapszuláció), és mivel `private` a listán belül, a listán kívüli programrész nem is látja, nem is tud a létezéséről.
  - Csak a belső szerkezet megvalósítására használjuk.
- A `Node`-nak minden adattagja `public`, ami azt jelenti, hogy a lista bármikor láthatja a `Node` belsejét.

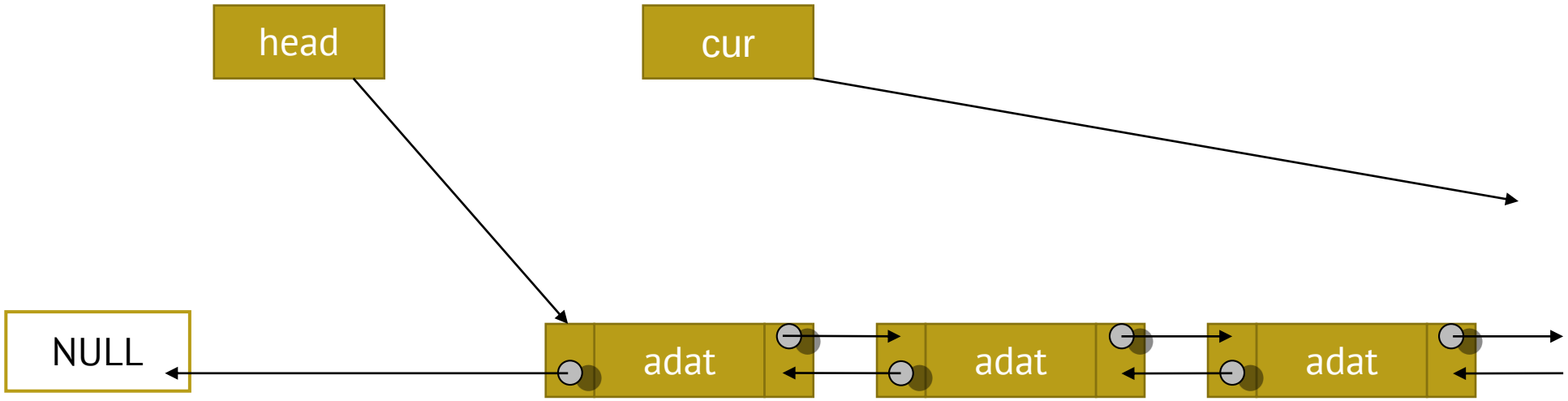
Láncolt lista elemei

- Három kitüntetett elem van.
  - A lista eleje, a vége és egy 'aktuális' elem.
- Ezzel az aktuális elemmel járhatjuk be az egész listát.

```
class List {
private:
    Node *head; // A lista eleje
    Node *tail; // A lista vége
    Node *cur;  // Az aktuális elem
};
```

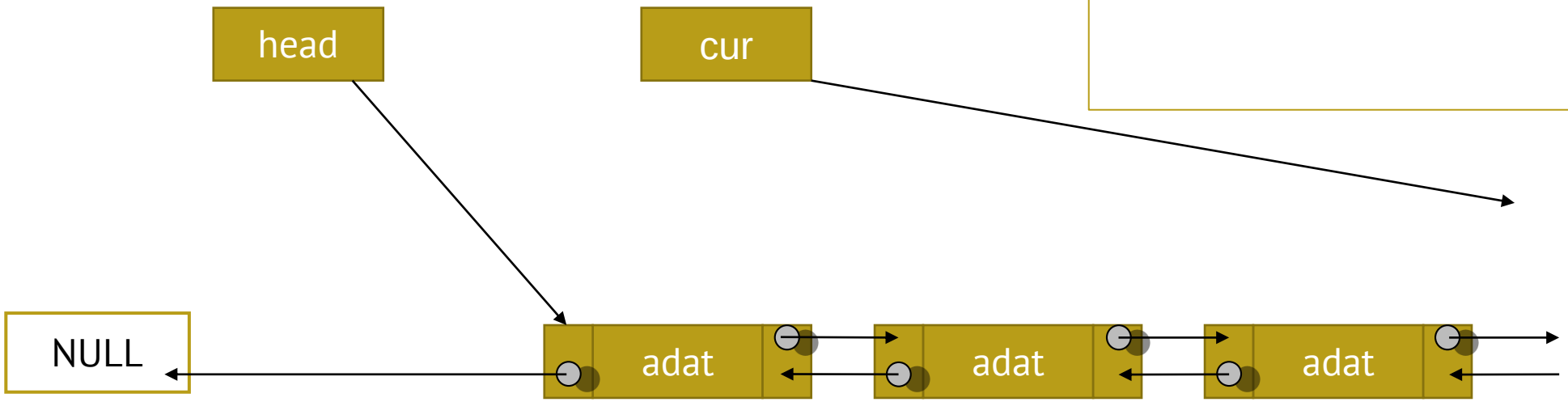
Láncolt lista – insertFirst

Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

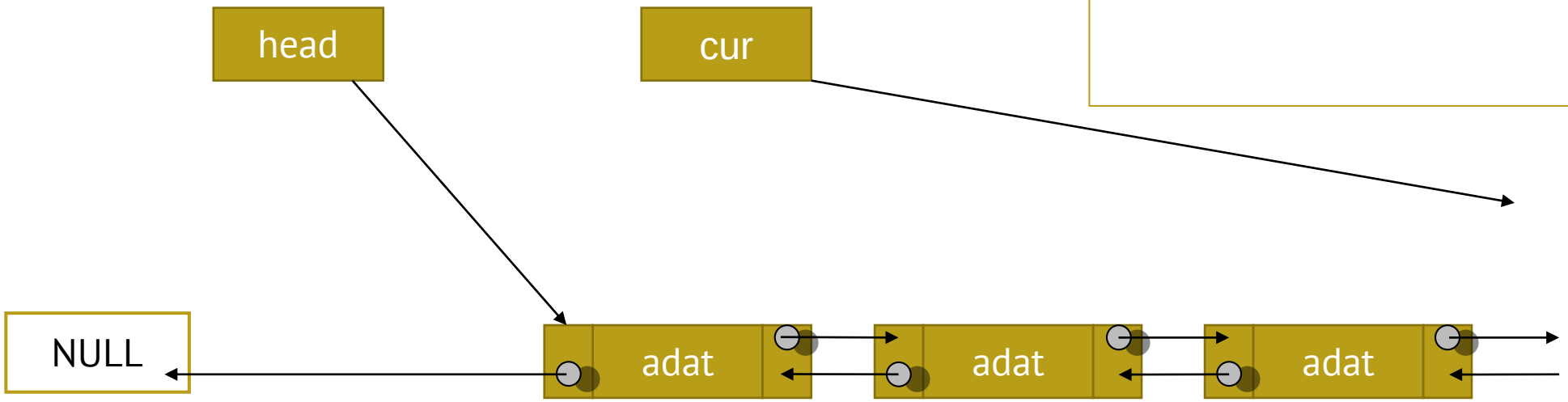
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
```

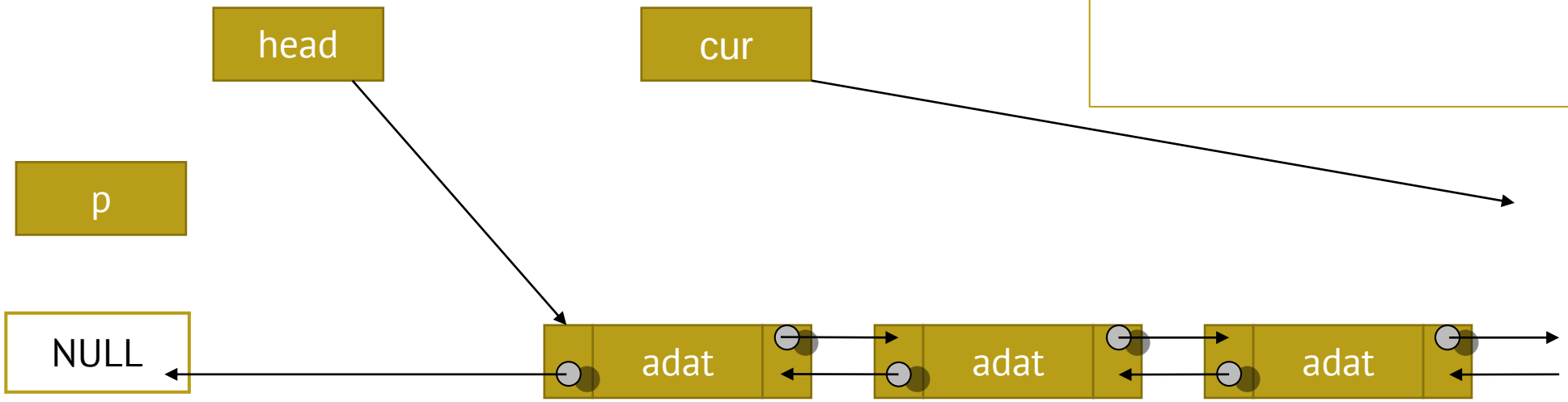
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
```

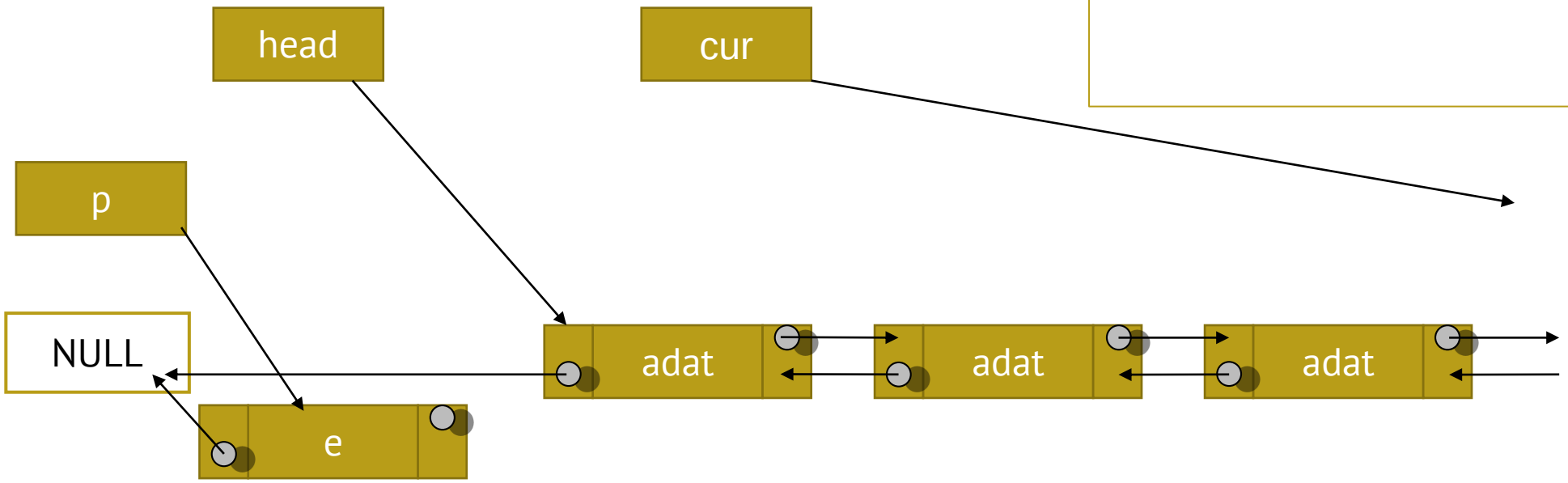
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
```

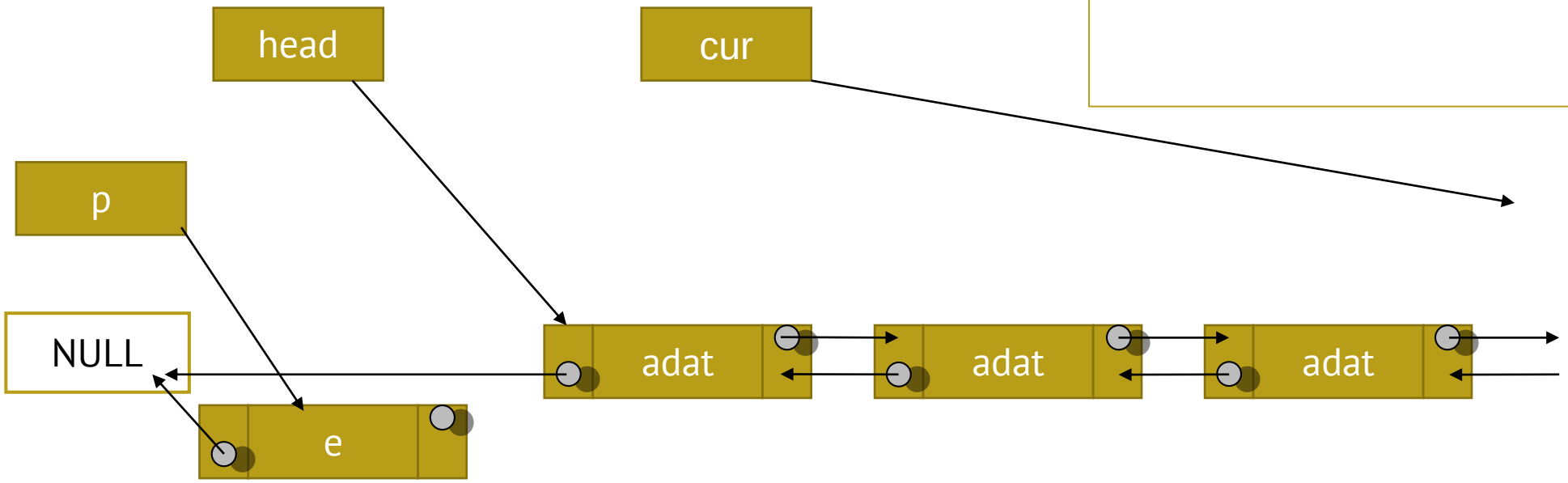
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
```

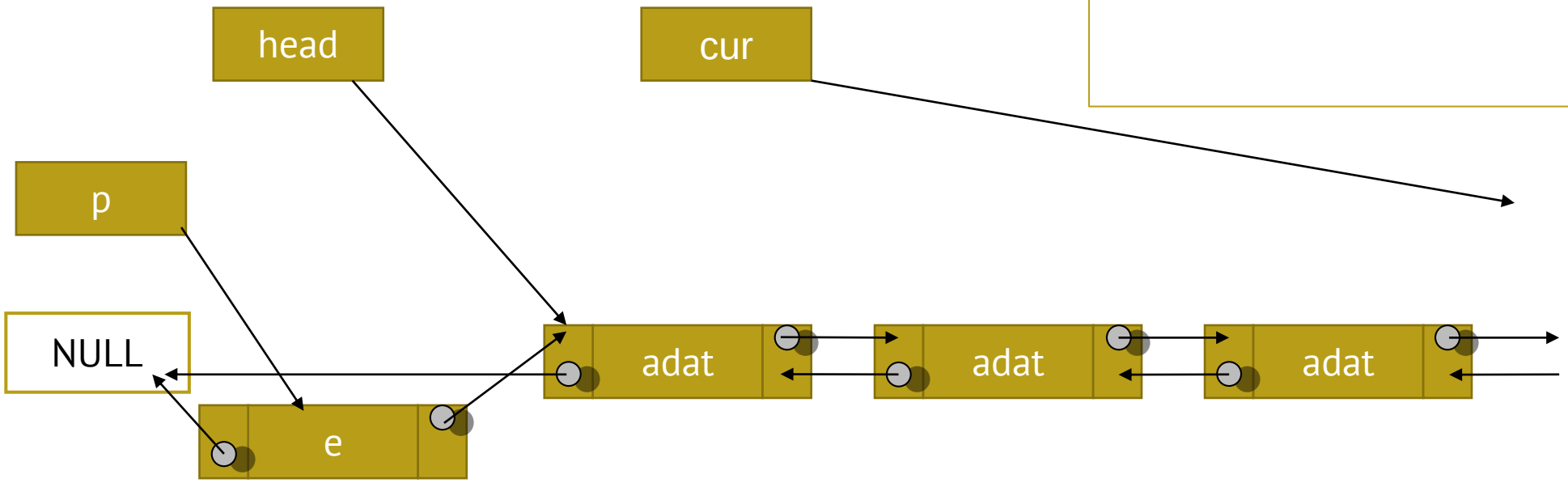
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
```

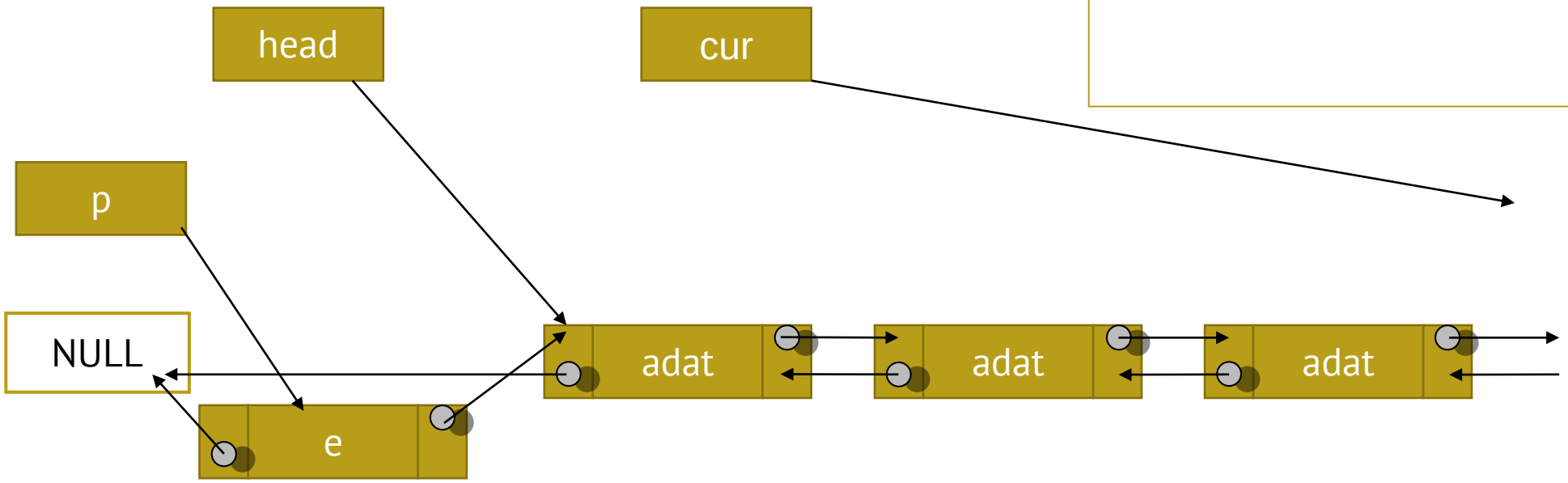
Példa 1: **insertFirst**(T e)



Láncolt lista – insertFirst

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
  head->prev = p;
```

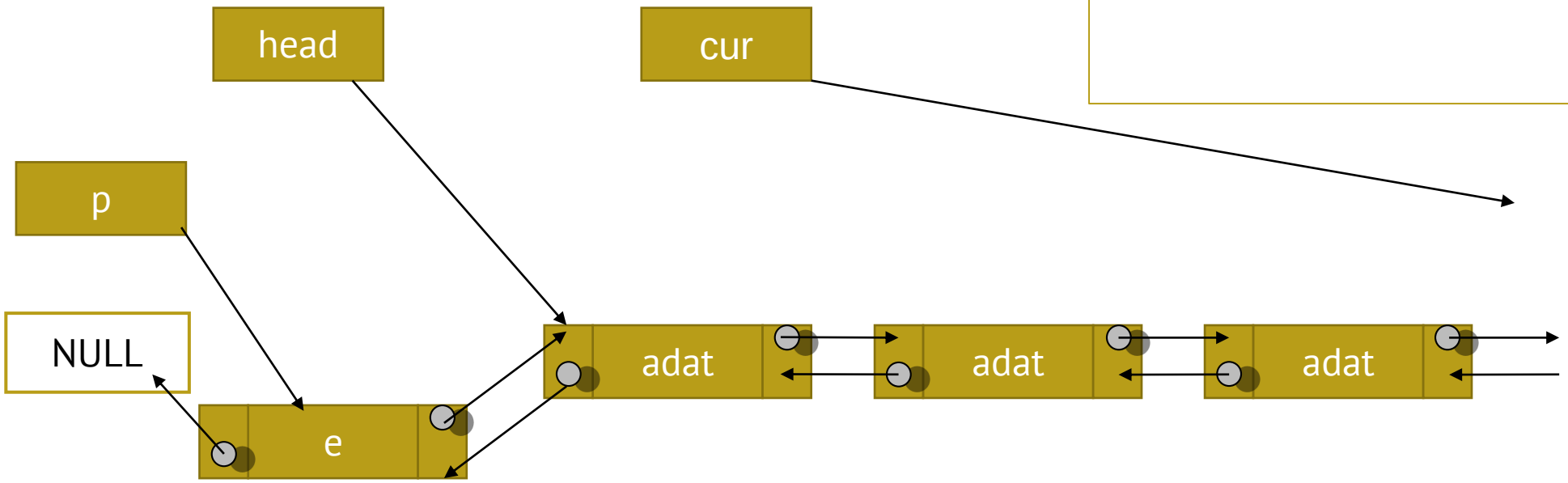
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
  head->prev = p;
```

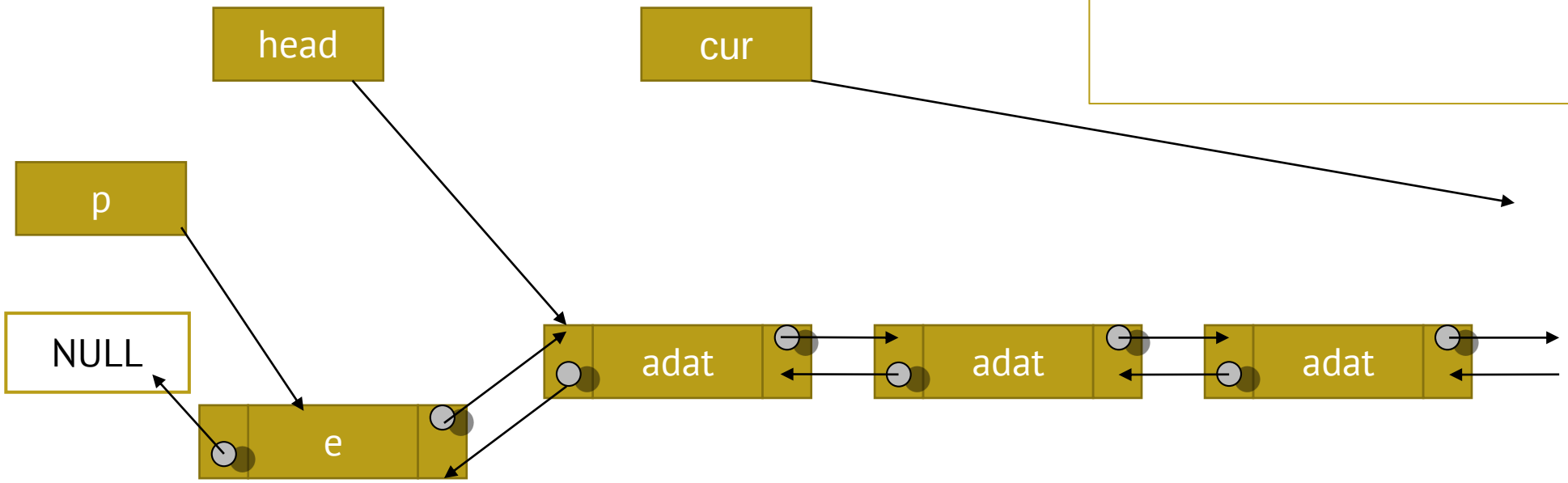
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
  head->prev = p;
  cur = head = p;
}
```

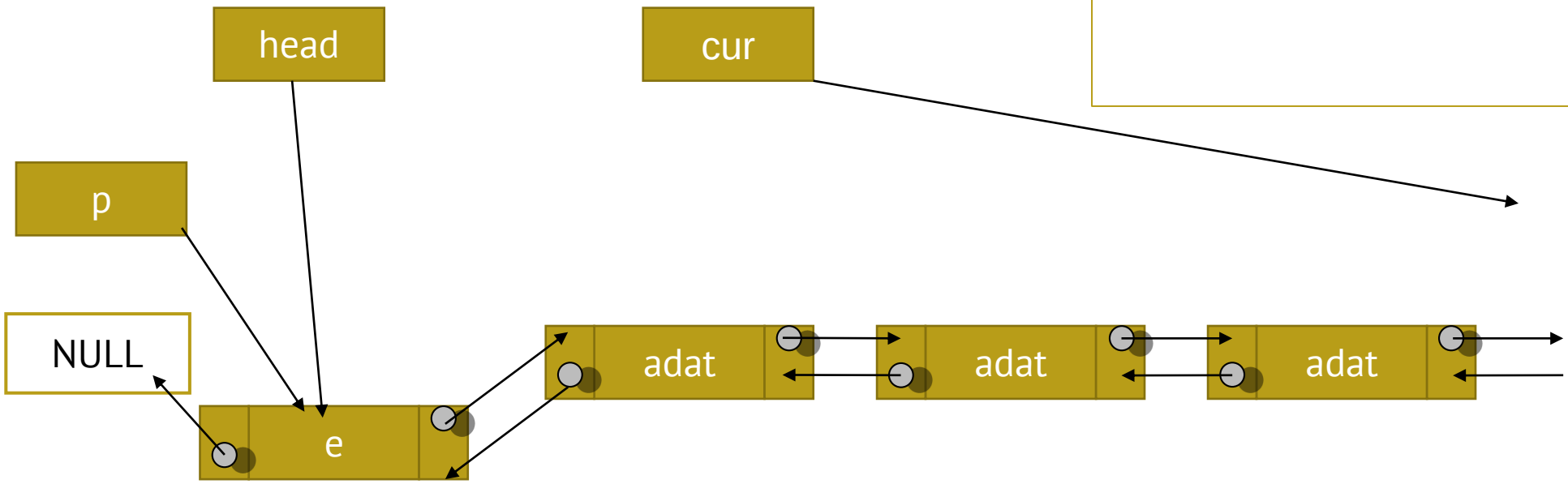
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
  head->prev = p;
  cur = head = p;
}
```

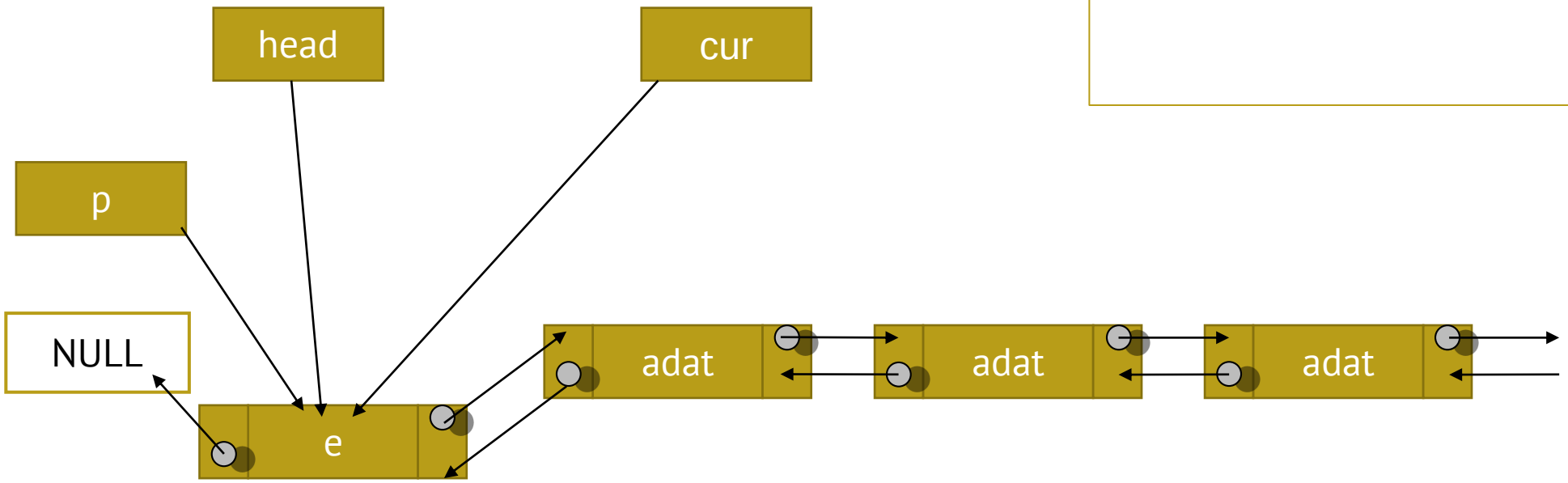
Példa 1: **`insertFirst`**`(T e)`



Láncolt lista – insertFirst

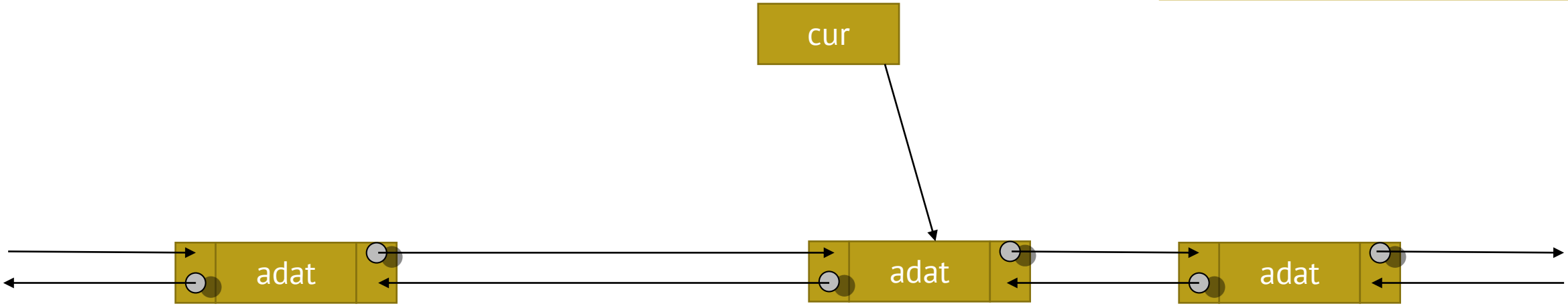
Példa 1: **`insertFirst`**`(T e)`

```
Node* p = new Node(e);
if (isEmpty()) {
  cur = head = tail = p;
} else {
  p->next = head;
  head->prev = p;
  cur = head = p;
}
```



Láncolt lista – insertBefore

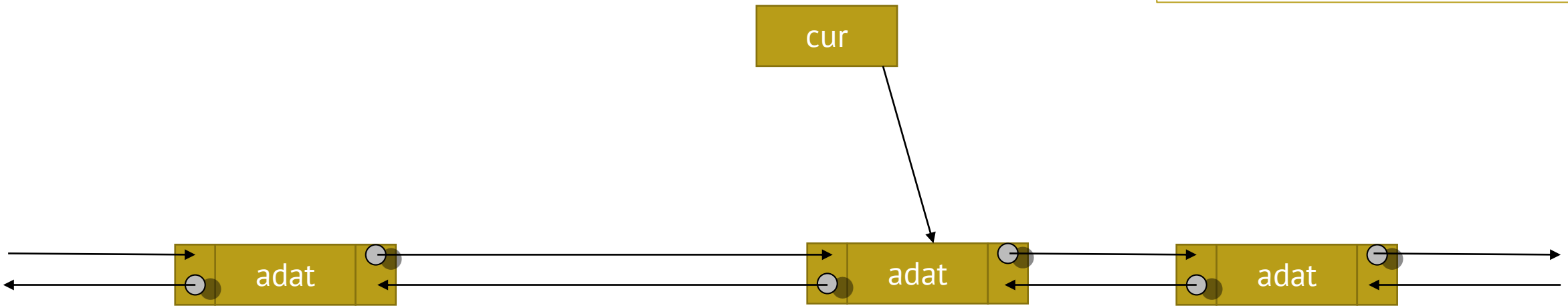
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
  e, cur->prev, cur);
```

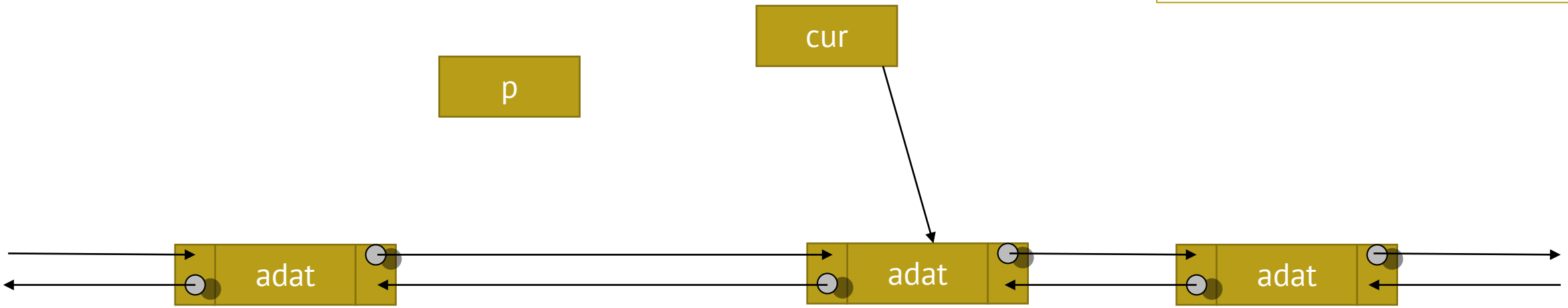
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
  e, cur->prev, cur);
```

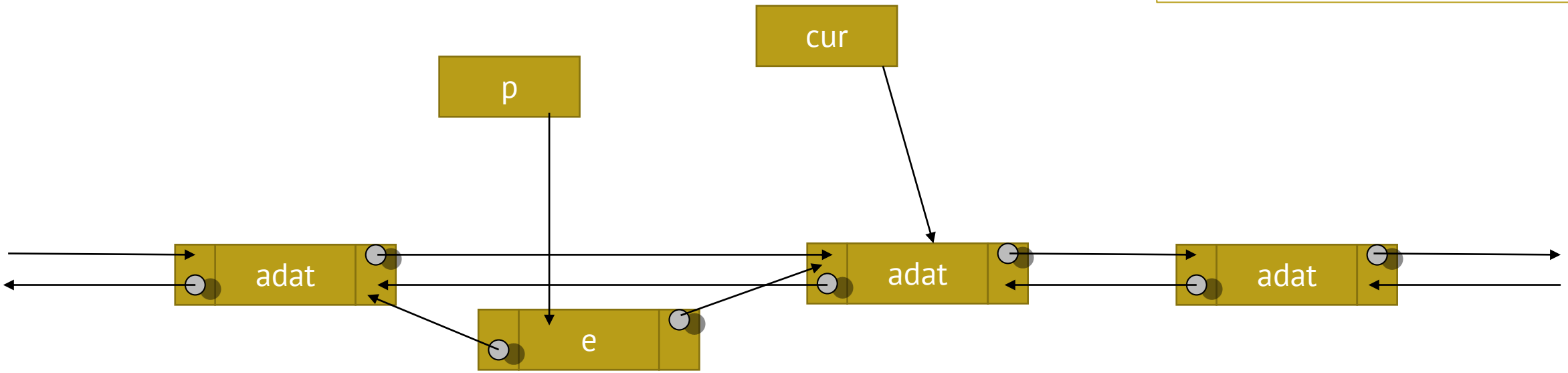
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
  e, cur->prev, cur);
```

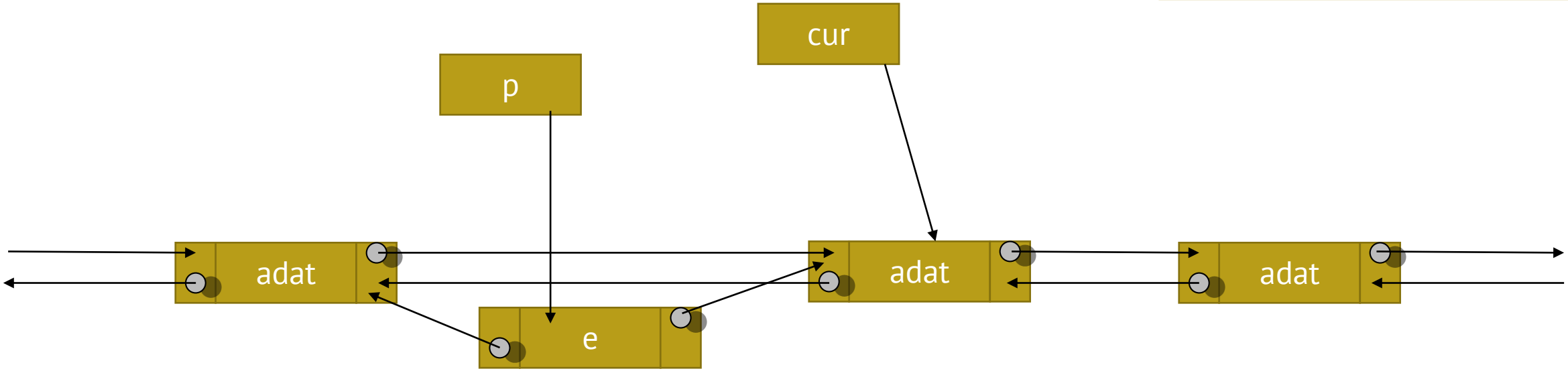
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
   e, cur->prev, cur);

cur->prev->next = p;
```

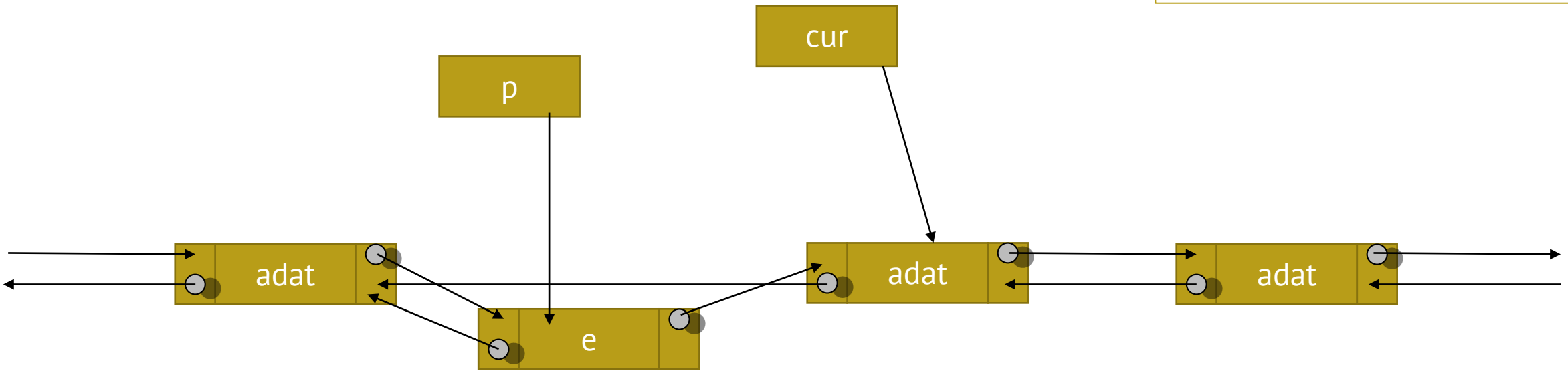
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
  e, cur->prev, cur);

cur->prev->next = p;
```

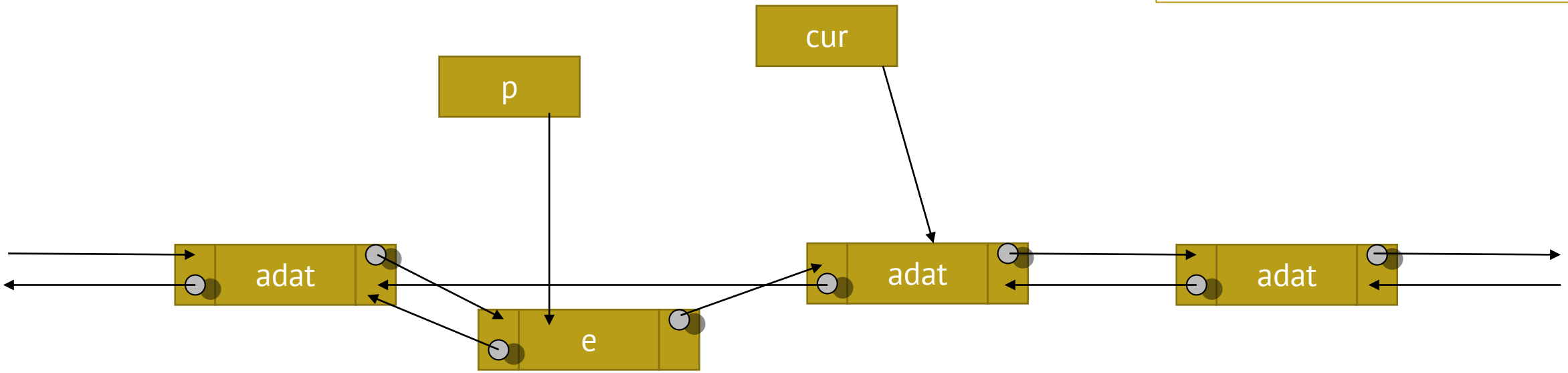
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
   e, cur->prev, cur);

cur->prev->next = p;
cur->prev = p;
```

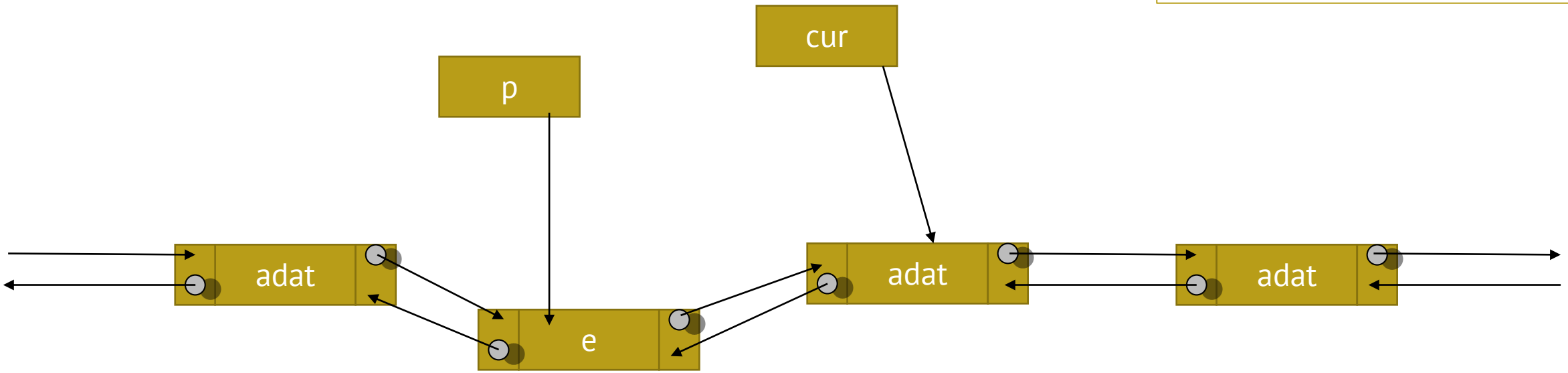
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
  e, cur->prev, cur);

cur->prev->next = p;
cur->prev = p;
```

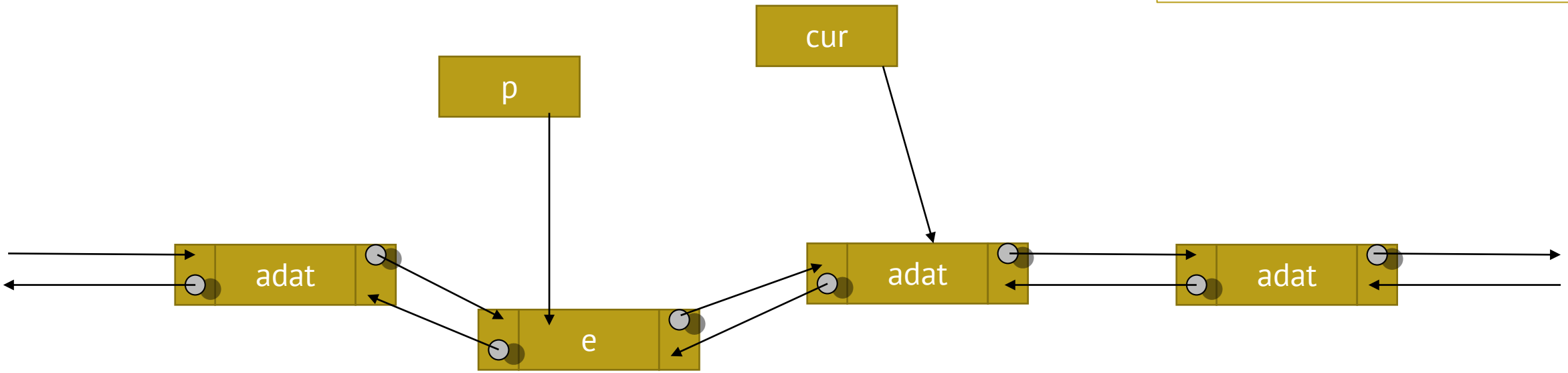
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

```
Node* p = new Node(
  e, cur->prev, cur);

cur->prev->next = p;
cur->prev = p;
cur = p;
```

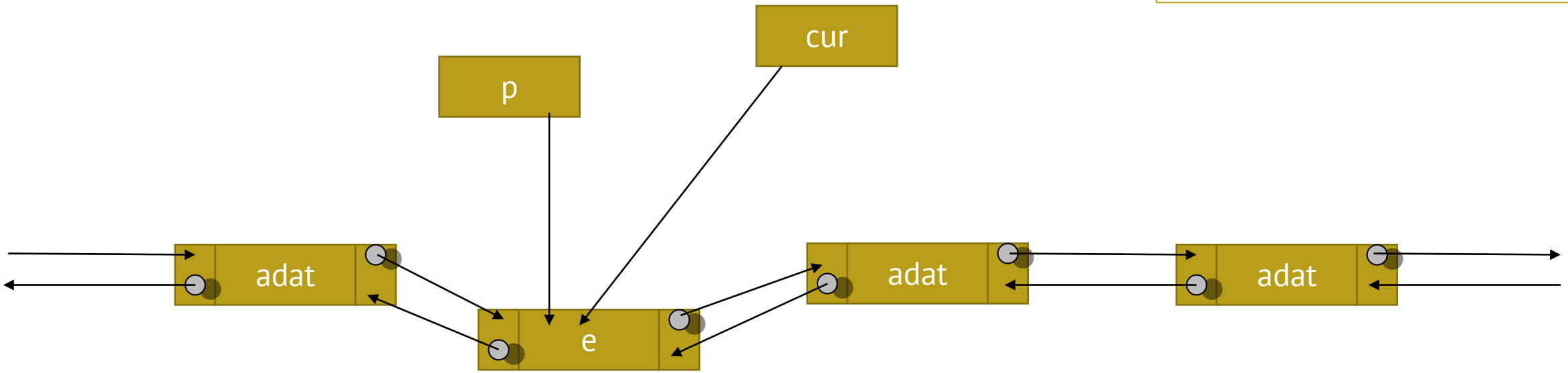
Példa 2: **`insertBefore`**`(T e)`



Láncolt lista – insertBefore

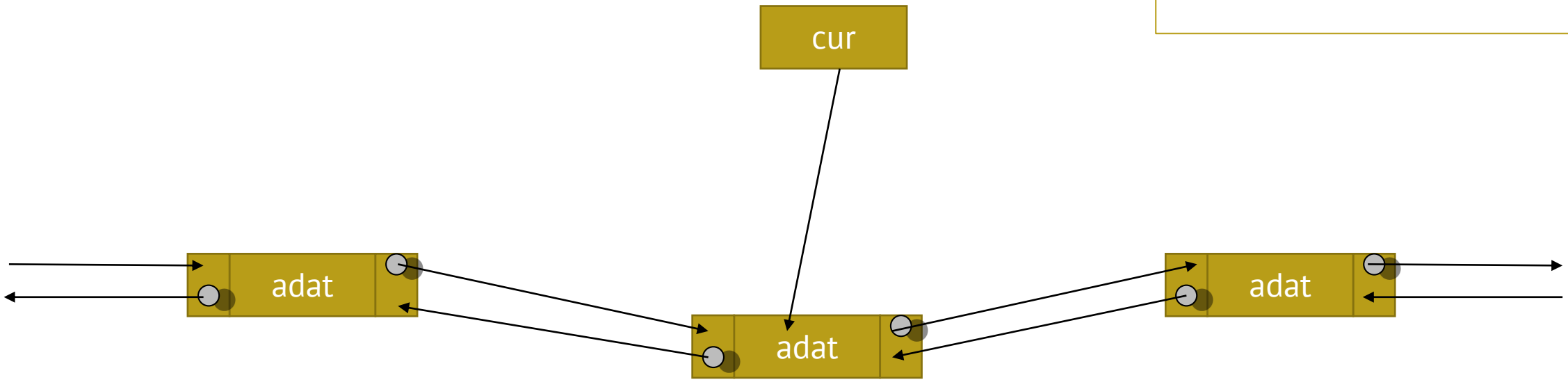
```
Node* p = new Node(
   e, cur->prev, cur);

cur->prev->next = p;
cur->prev = p;
cur = p;
```

Példa 2: **`insertBefore`**`(T e)`



Láncolt lista - removeCur

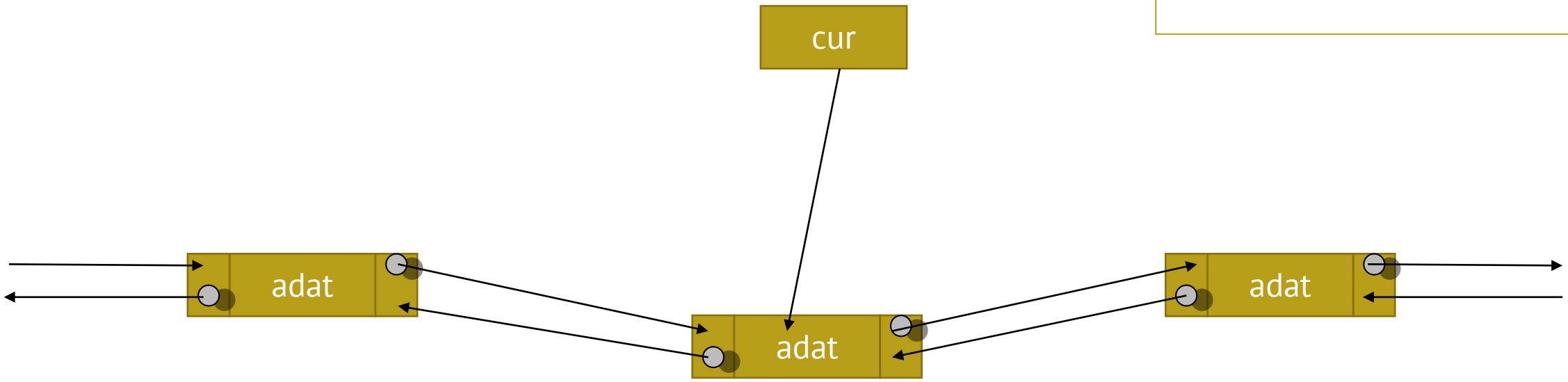
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;
```

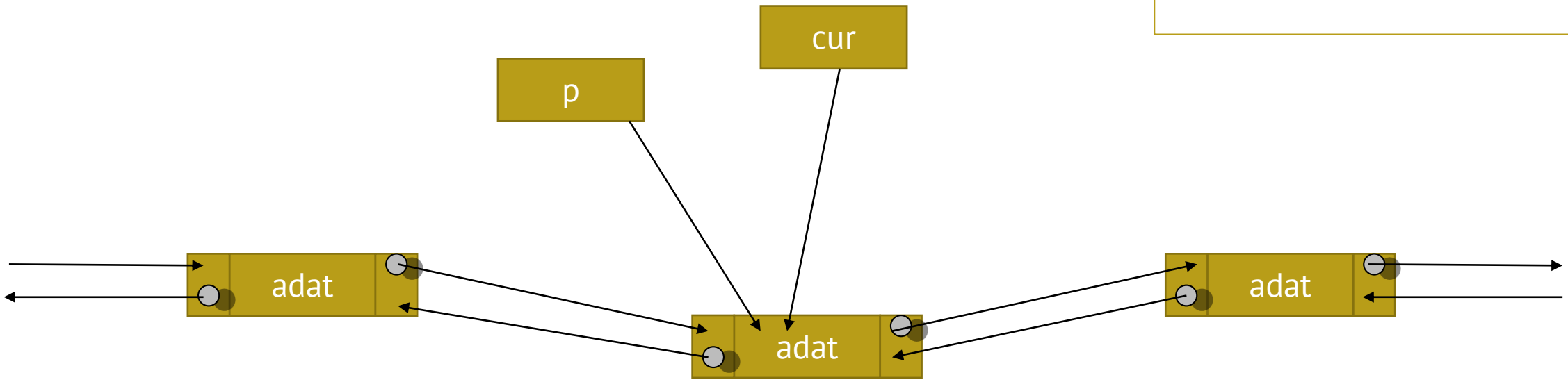
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;
```

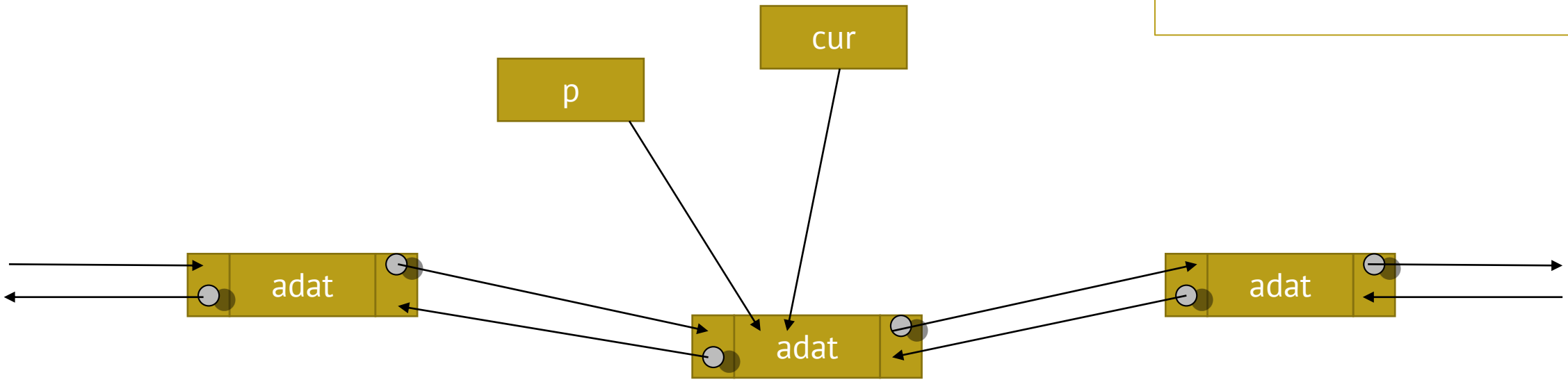
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
```

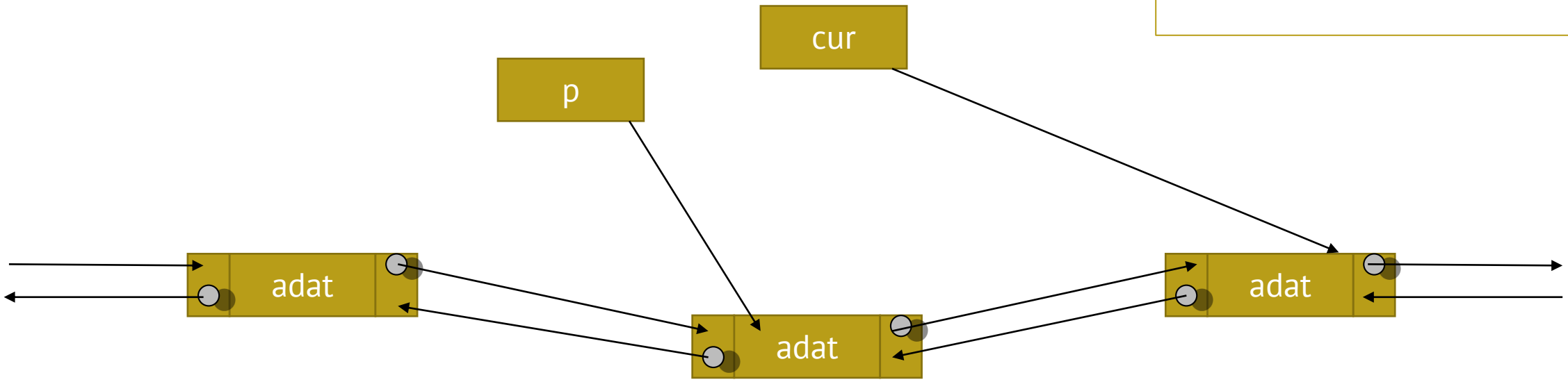
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
```

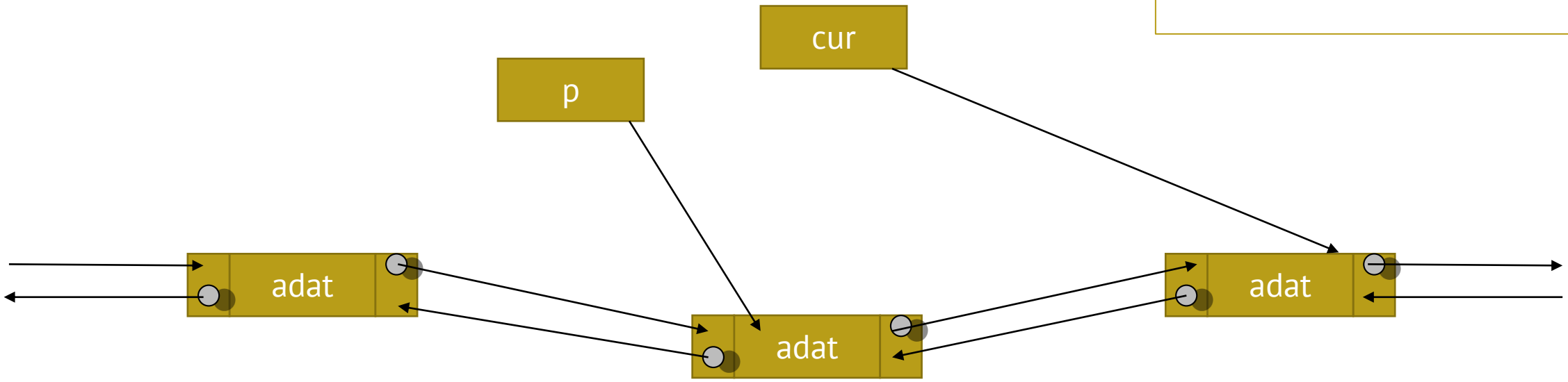
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
cur->prev = p->prev;
```

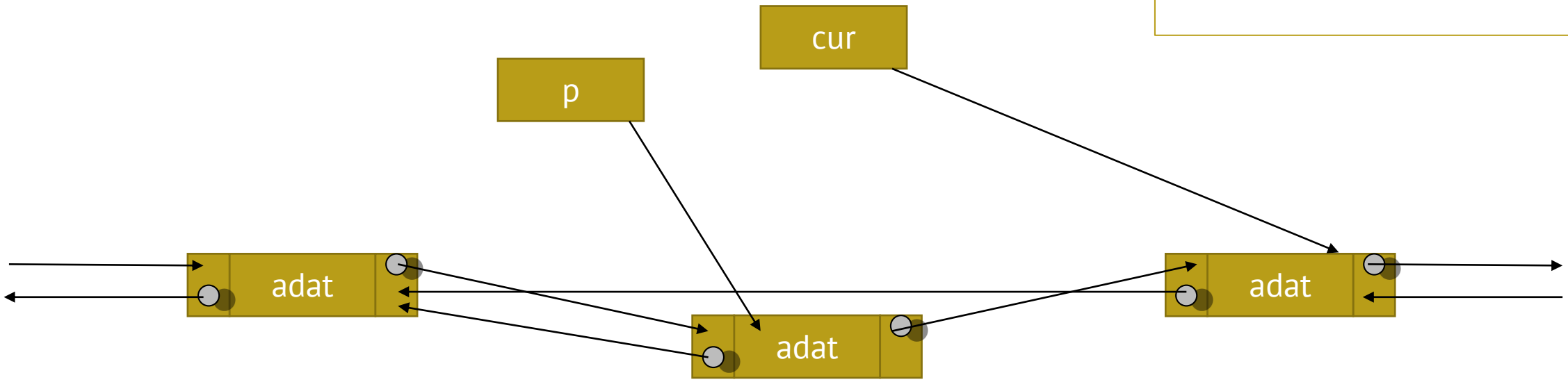
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
cur->prev = p->prev;
```

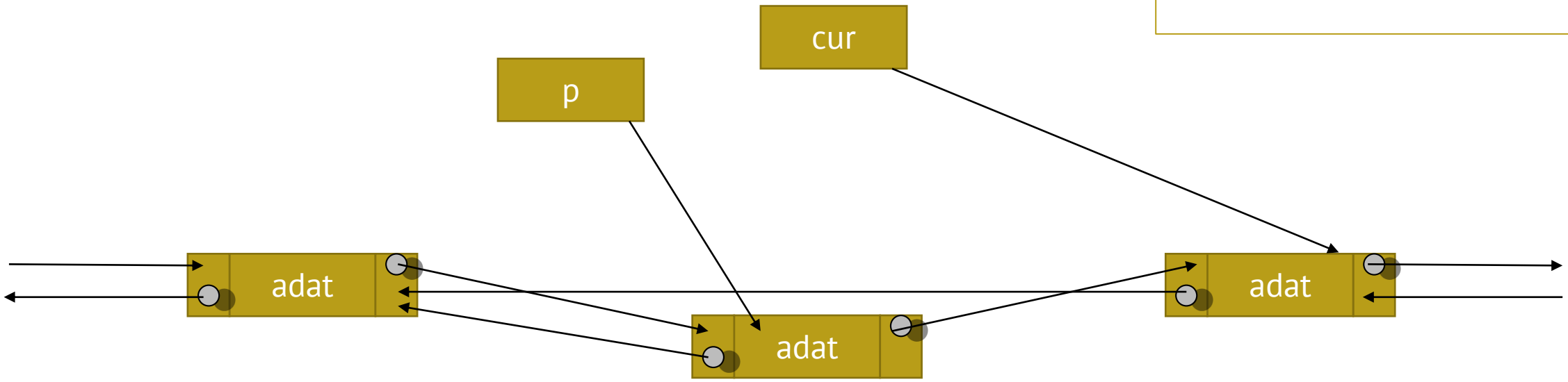
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
cur->prev = p->prev;
p->prev->next = cur;
```

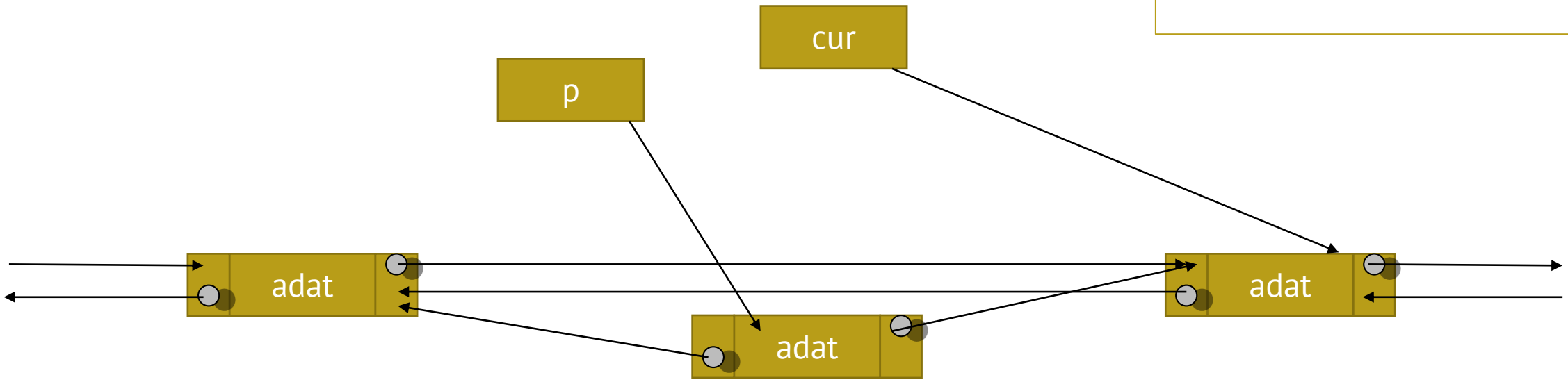
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
cur->prev = p->prev;
p->prev->next = cur;
```

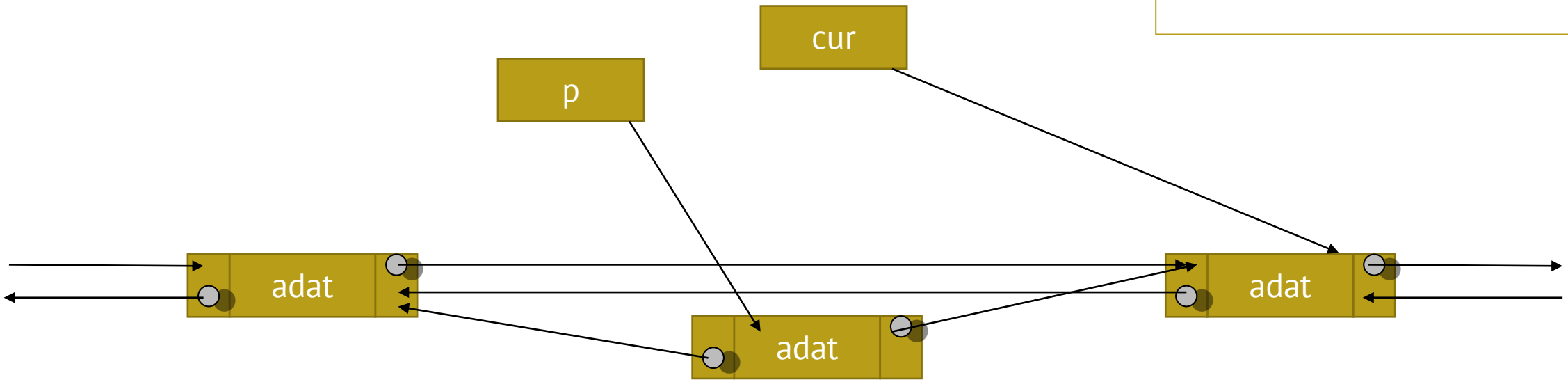
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
cur->prev = p->prev;
p->prev->next = cur;
delete p;
```

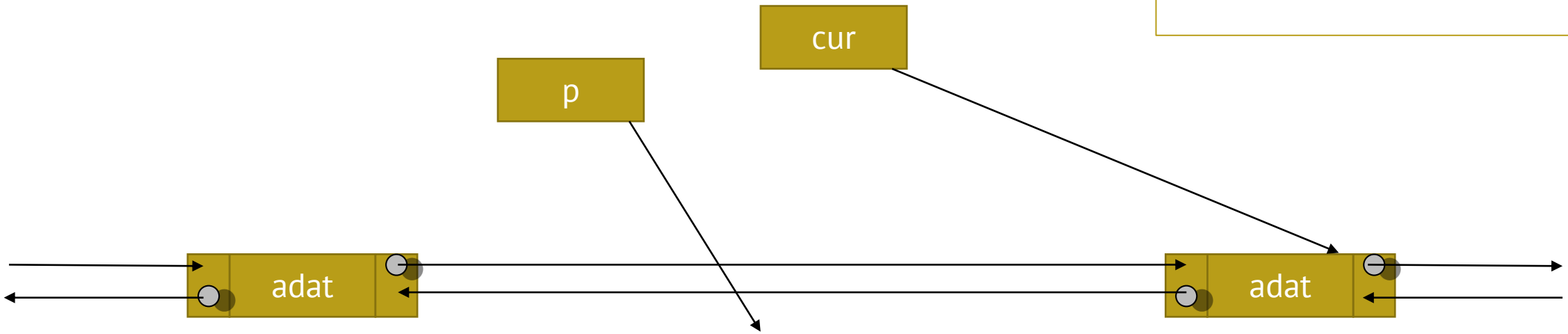
Példa 3: **`removeCur()`**



Láncolt lista - removeCur

```
Node* p = cur;

cur = cur->next;
cur->prev = p->prev;
p->prev->next = cur;
delete p;
```

Példa 3: **`removeCur()`**



Sor és Láncolt Lista

Következő téma