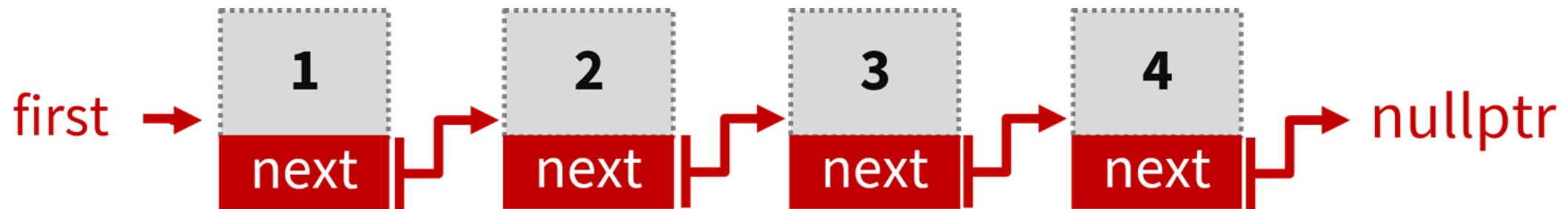


Zret'azený zoznam



OBSAH

- Zreťazený zoznam
- Reprezentácia v C/C++

OBSAH

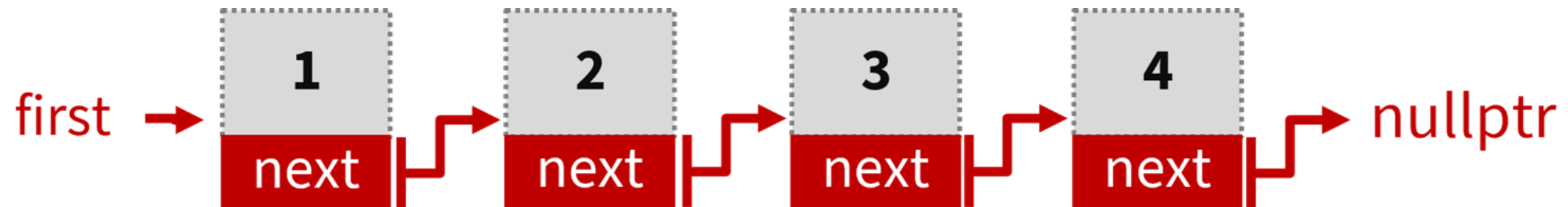
- Zret'azený zoznam
- Reprezentácia v C/C++

Základné operácie

- pridávanie uzlov (začiatok, koniec, do stredu)
- prechod zoznamom (výpis, hľadanie)
- vymazávanie uzlov (začiatok, koniec, v strede)

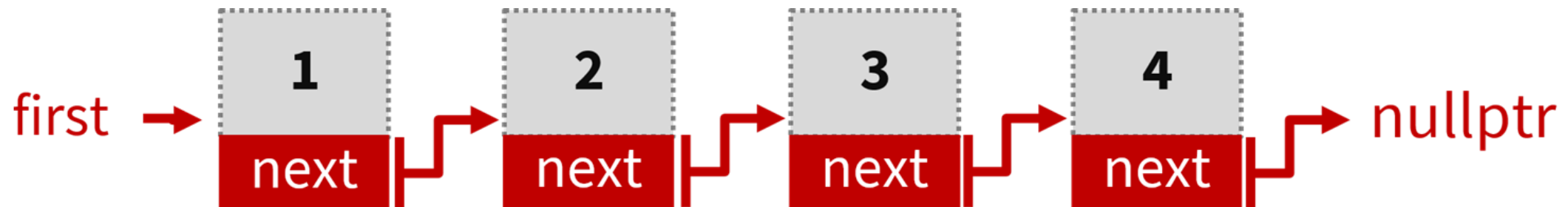
Zret'azený zoznam

- Dátová štruktúra obsahujúca navzájom prepojené uzly v pamäti (nie sú uložené spojito za sebou)



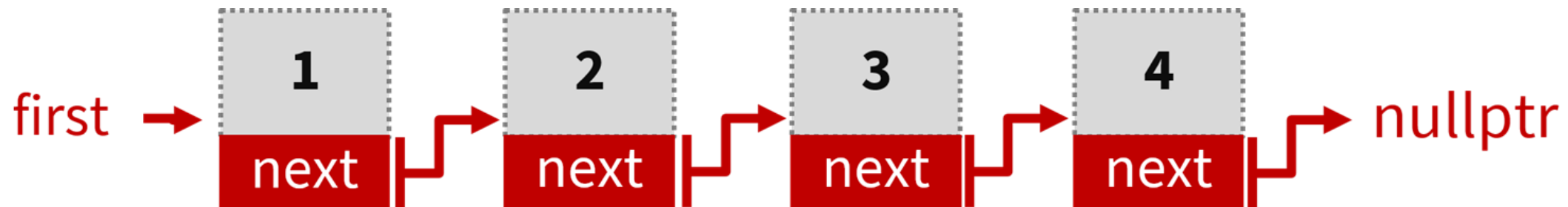
Zret'azený zoznam

- Dátová štruktúra obsahujúca navzájom prepojené uzly v pamäti (nie sú uložené spojitاً za sebou)
- Uzly sú prepojené pomocou smerníkov



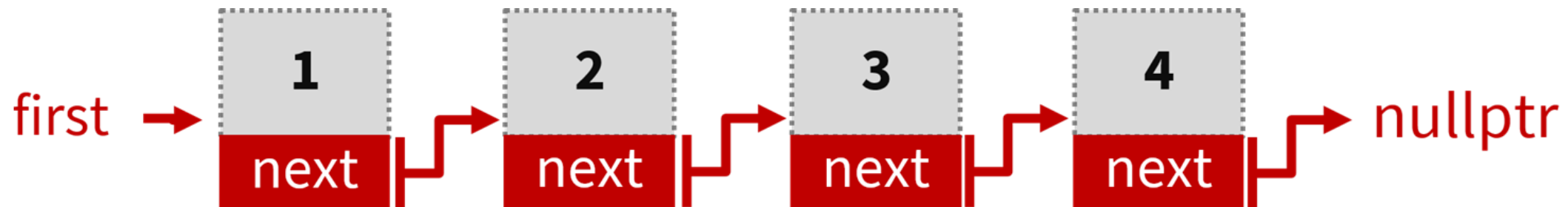
Zret'azený zoznam

- Dátová štruktúra obsahujúca navzájom prepojené uzly v pamäti (nie sú uložené spojitاً za sebou)
- Uzly sú prepojené pomocou smerníkov
- Lineárna dátová štruktúra (sekvenčný prístup)



Zret'azený zoznam

- Dátová štruktúra obsahujúca navzájom prepojené uzly v pamäti (nie sú uložené spojito za sebou)
- Uzly sú prepojené pomocou smerníkov
- Lineárna dátová štruktúra (sekvenčný prístup)
- Dynamická dátová štruktúra (premenlivý počet uzlov)



Zret'azený zoznam

- **Ideálne využitie:** v prípadoch, kedy dopredu nepoznáme počet prvkov (pridávanie/odoberanie prvkov za behu)

Zret'azený zoznam

- **Ideálne využitie:** v prípadoch, kedy dopredu nepoznáme počet prvkov (pridávanie/odoberanie prvkov za behu)

Výhody

- Výpočtovo nenáročné pridávanie/vymazávanie uzlov
- Lepšie využitie pamäte ako pole (v pamäti je vyhradené miesto pre aktuálny počet uzlov)

Zret'azený zoznam

Nevýhody

- Sekvenčný prístup (pomalé)
- 1 uzol zoznamu zaberá viac pamäte ako prvok v poli (uzol okrem dát obsahuje aj smerník na nasledovníka)

Zret'azený zoznam

Typy zoznamov

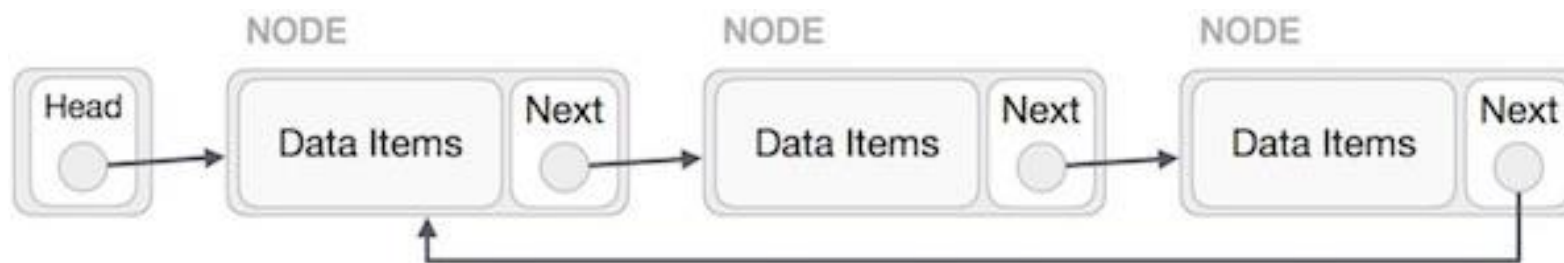
- Jednosmerný
- Jednosmerný cirkulárny
- Obojsmerný
- Obojsmerný cirkulárny



Zret'azený zoznam

Typy zoznamov

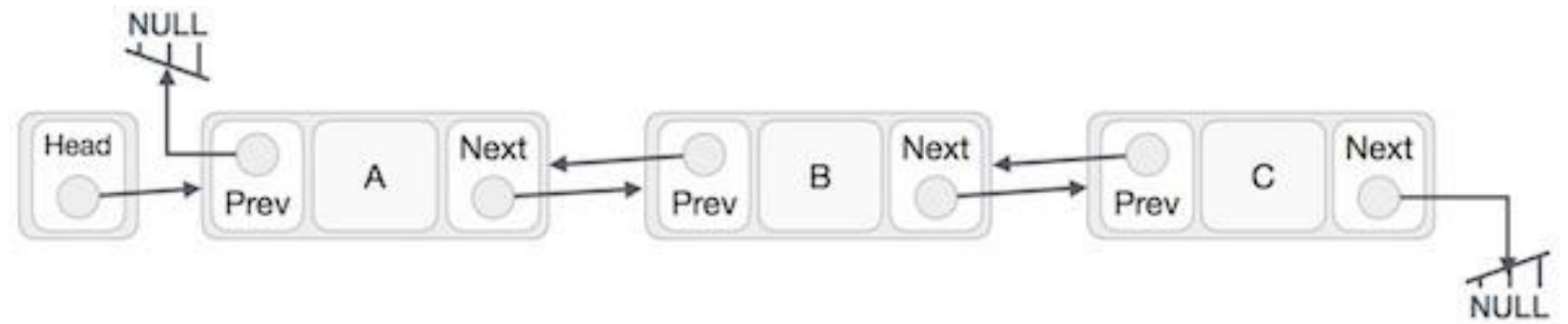
- Jednosmerný
- **Jednosmerný cirkulárny**
- Obojsmerný
- Obojsmerný cirkulárny



Zret'azený zoznam

Typy zoznamov

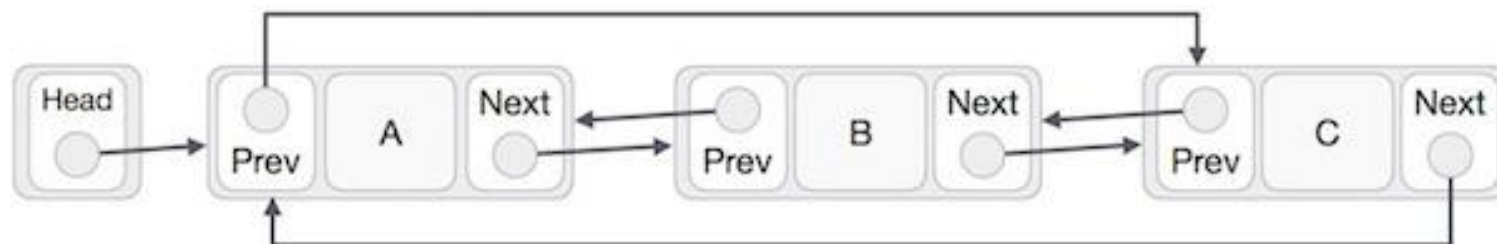
- Jednosmerný
- Jednosmerný cirkulárny
- **Obojsmerný**
- Obojsmerný cirkulárny



Zret'azený zoznam

Typy zoznamov


- Jednosmerný
- Jednosmerný cirkulárny
- Obojsmerný
- Obojsmerný cirkulárny



Jednosmerne zret'azený zoznam

first →

Jednosmerne zret'azený zoznam

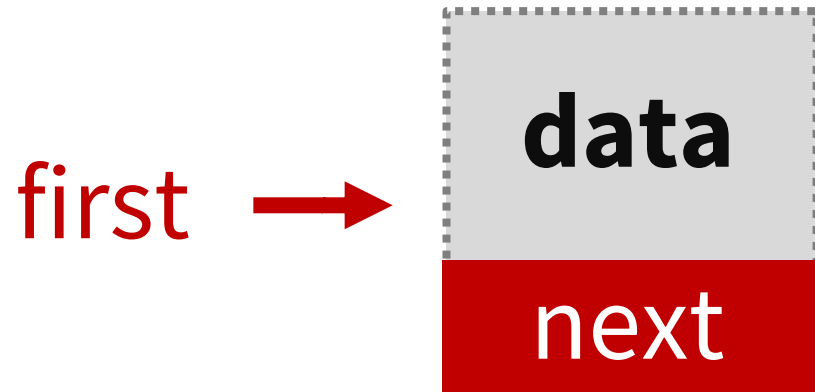


Smerník na
prvý uzol
zoznamu

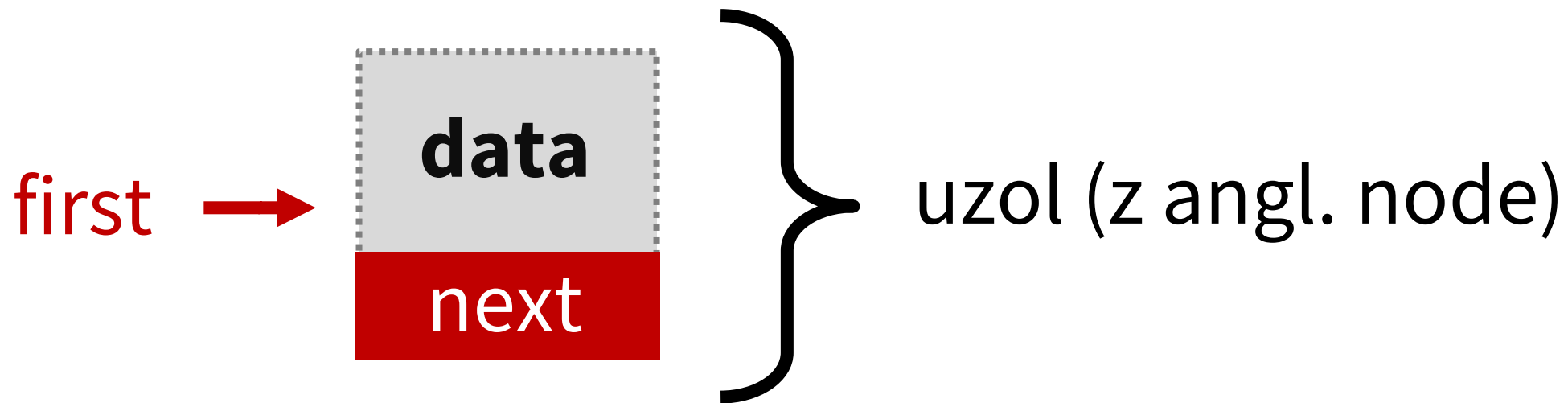
The diagram illustrates a singly linked list. A red arrow points from the word 'first' to the right. Above this arrow, a green speech bubble contains the text 'Smerník na prvý uzol zoznamu'.

first →

Jednosmerne zret'azený zoznam



Jednosmerne zret'azený zoznam



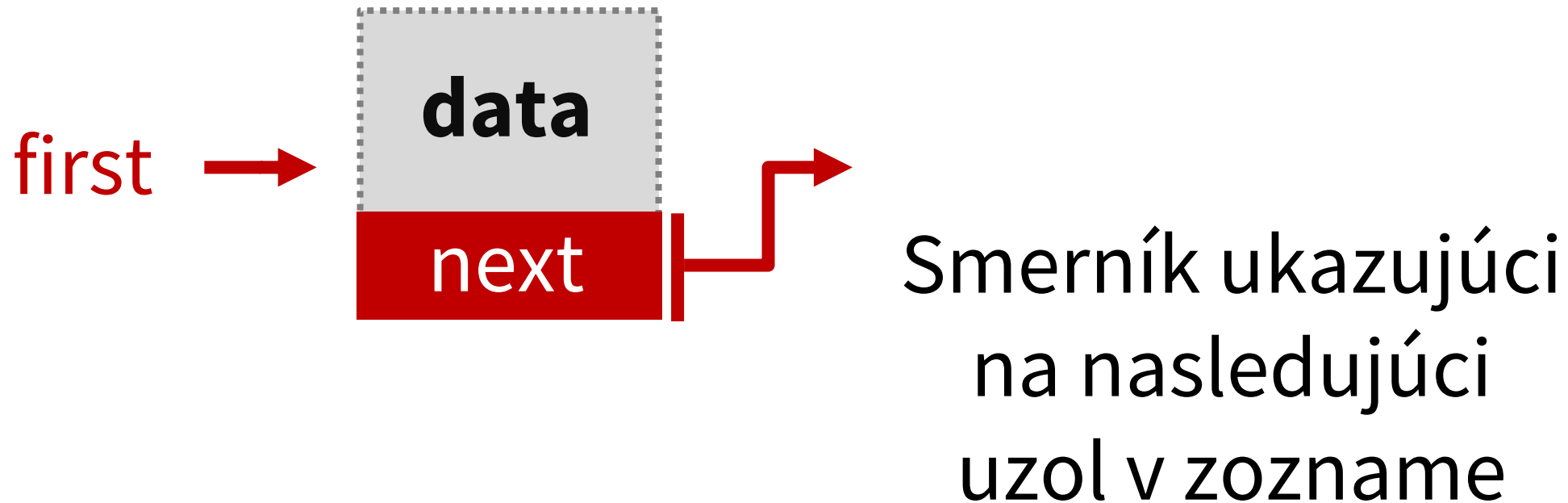
Jednosmerne zret'azený zoznam



Jednosmerne zret'azený zoznam



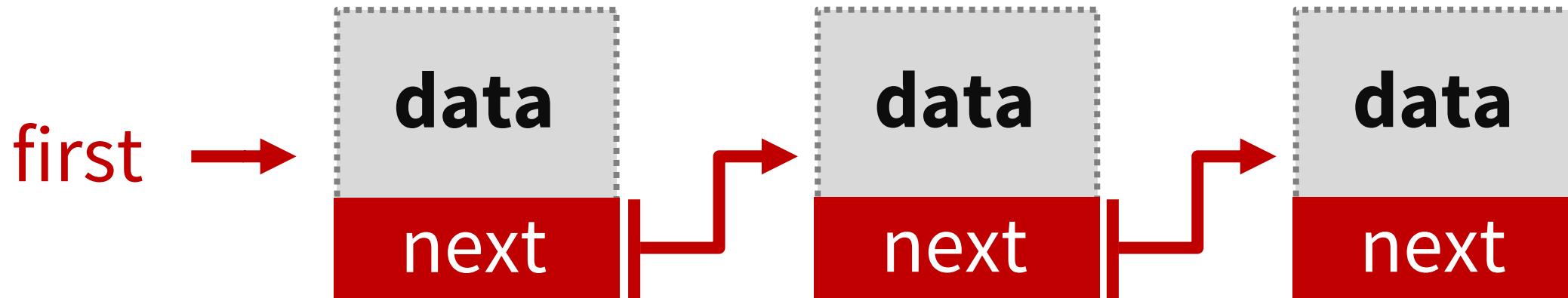
Jednosmerne zret'azený zoznam



Jednosmerne zret'azený zoznam



Jednosmerne zret'azený zoznam



Jednosmerne zret'azený zoznam

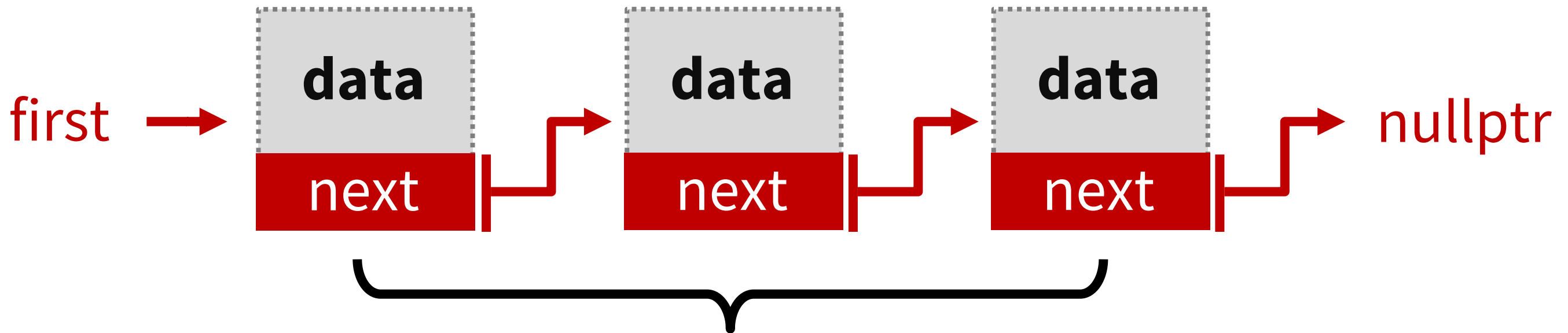


Jednosmerne zret'azený zoznam



Nasledovník posledného uzla
je nastavený na **nullptr**

Jednosmerne zret'azený zoznam



Uzly nemusia byť uložené v pamäti spojito

Reprezentácia v C/C++

Reprezentácia uzla

```
struct Node {  
    int data;  
    Node* next;  
};
```



Reprezentácia v C/C++

Reprezentácia uzla

```
struct Node {  
    int data;  
    Node* next;  
};
```



Reprezentácia celého zoznamu

```
struct List {  
    Node* first;  
};
```

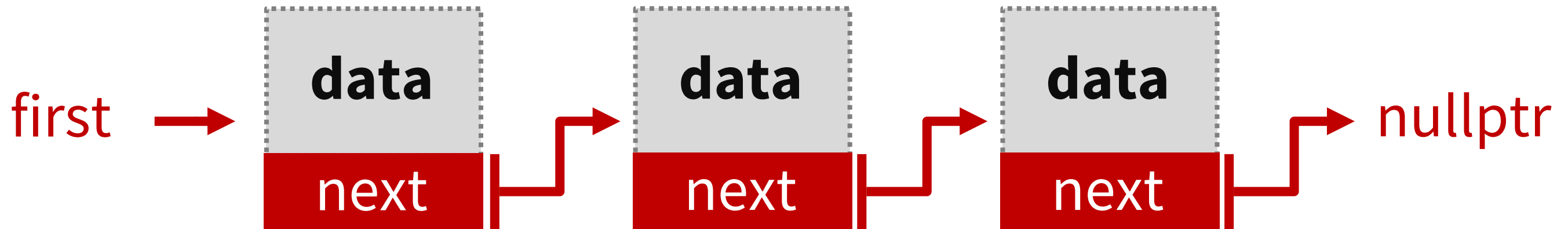


Reprezentácia v C/C++

```
struct Node {  
    int data;  
    Node* next;  
};
```



```
struct List {  
    Node* first;  
};
```



Operácie:

Pridanie uzla na začiatok ●

Pridanie uzla do stredu

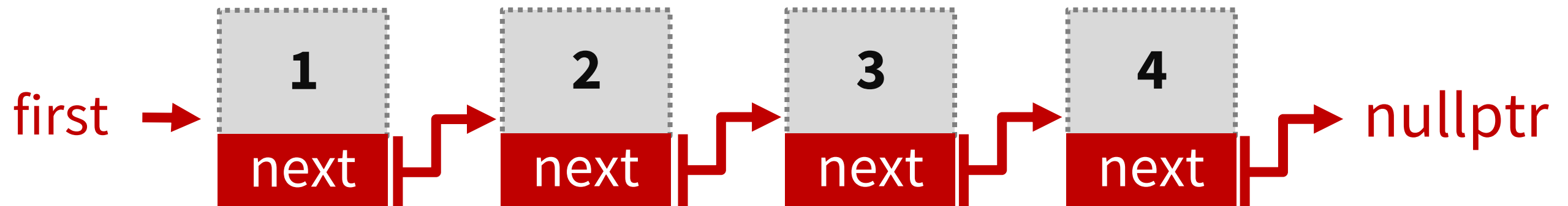
Pridanie uzla na koniec

Výpis zoznamu

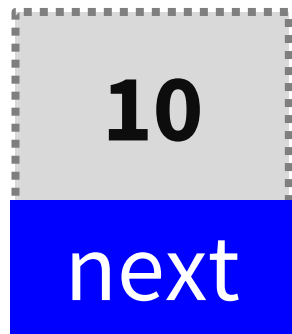
Vymazanie prvého uzla

Vymazanie i-teho uzla

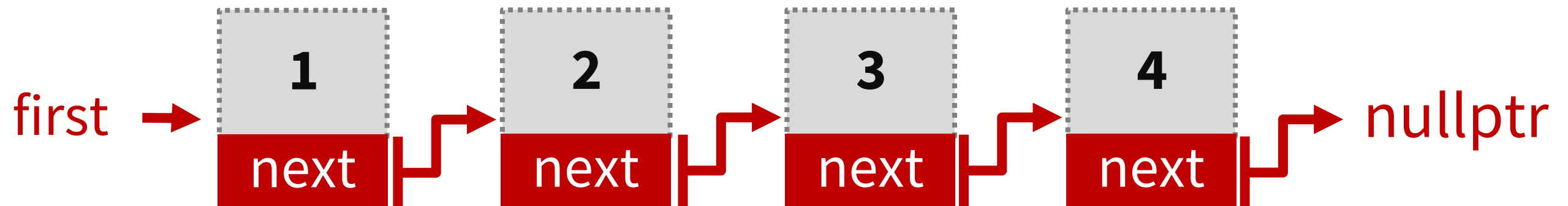
Pridanie uzla na začiatok



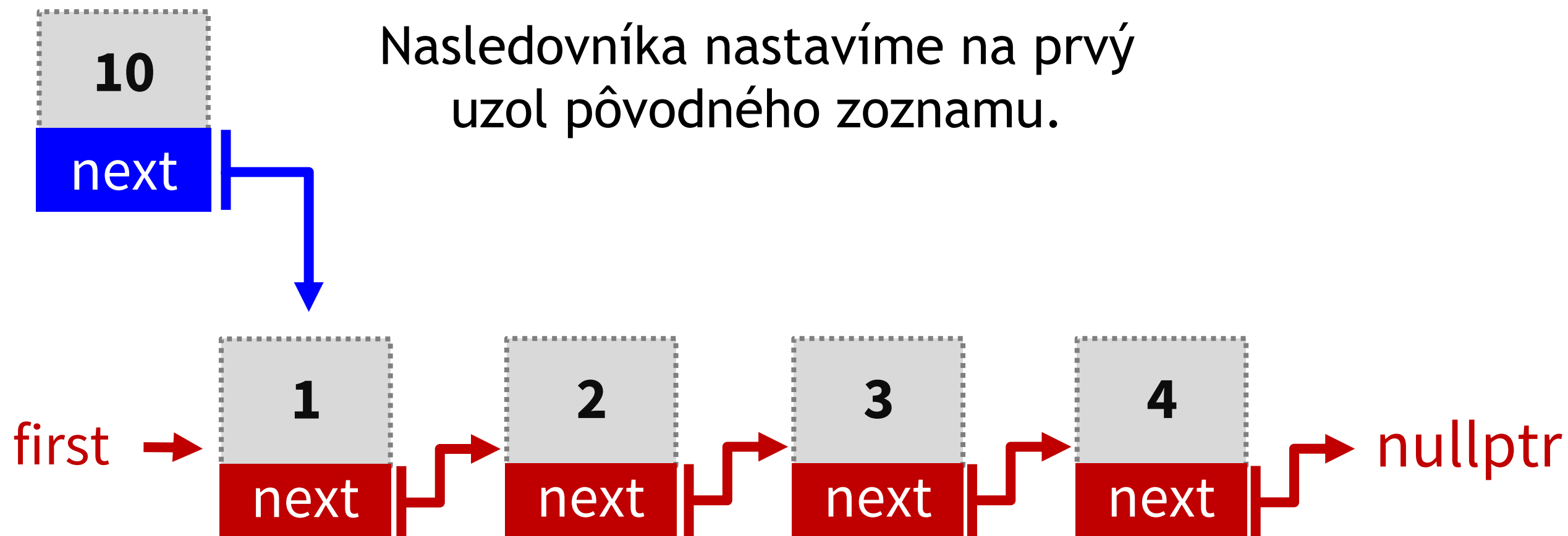
Pridanie uzla na začiatok



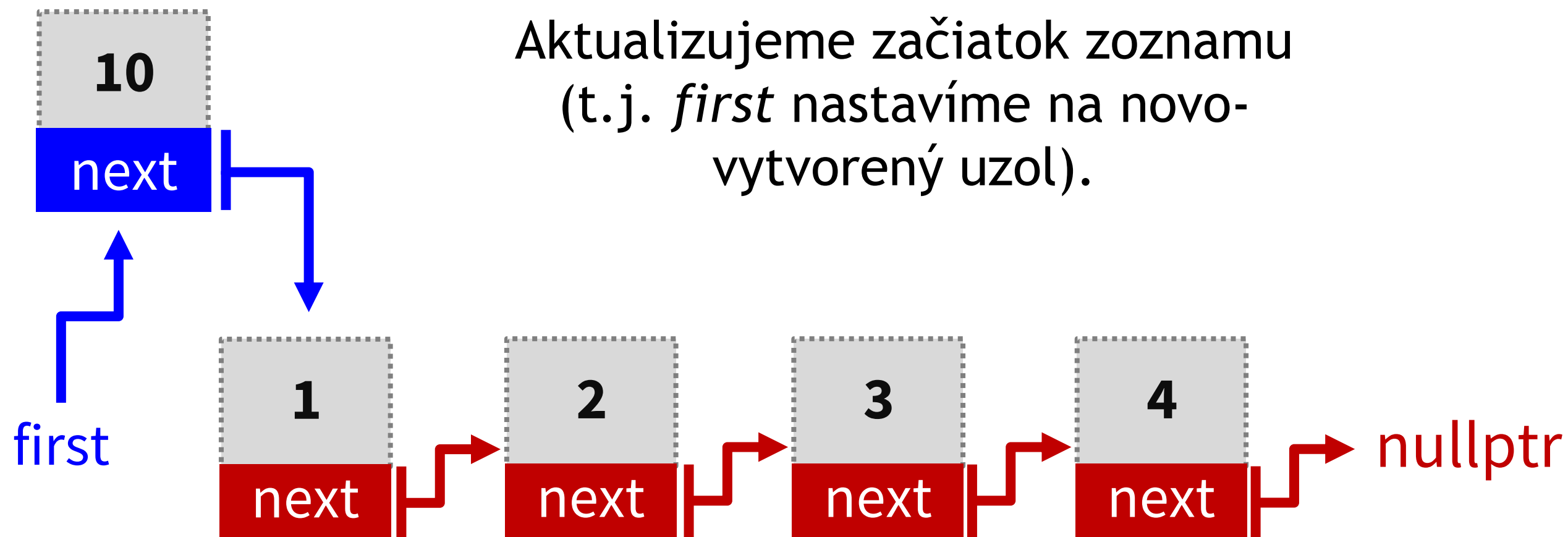
Vytvoríme nový uzol



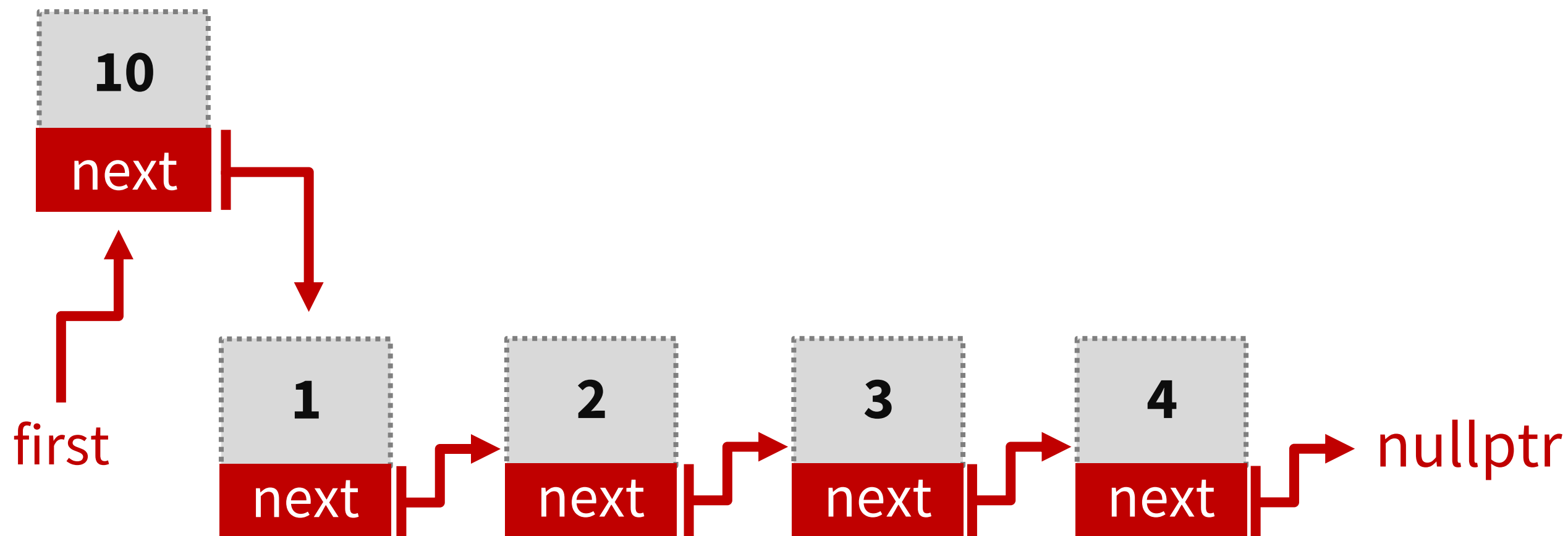
Pridanie uzla na začiatok



Pridanie uzla na začiatok



Pridanie uzla na začiatok



Operácie:

Pridanie uzla na začiatok

Pridanie uzla do stredu ●

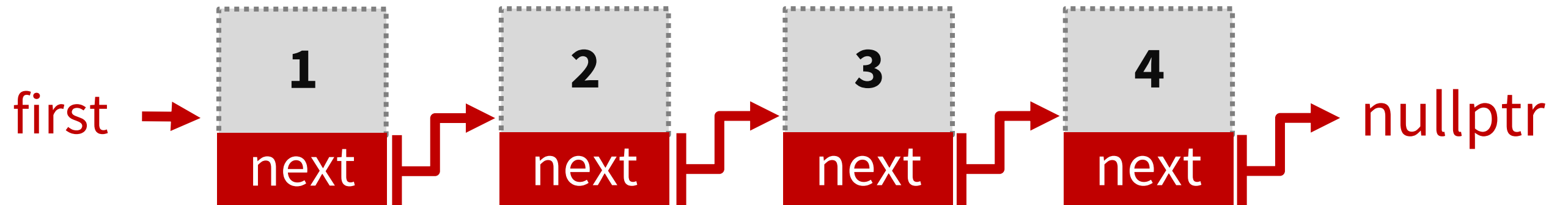
Pridanie uzla na koniec

Výpis zoznamu

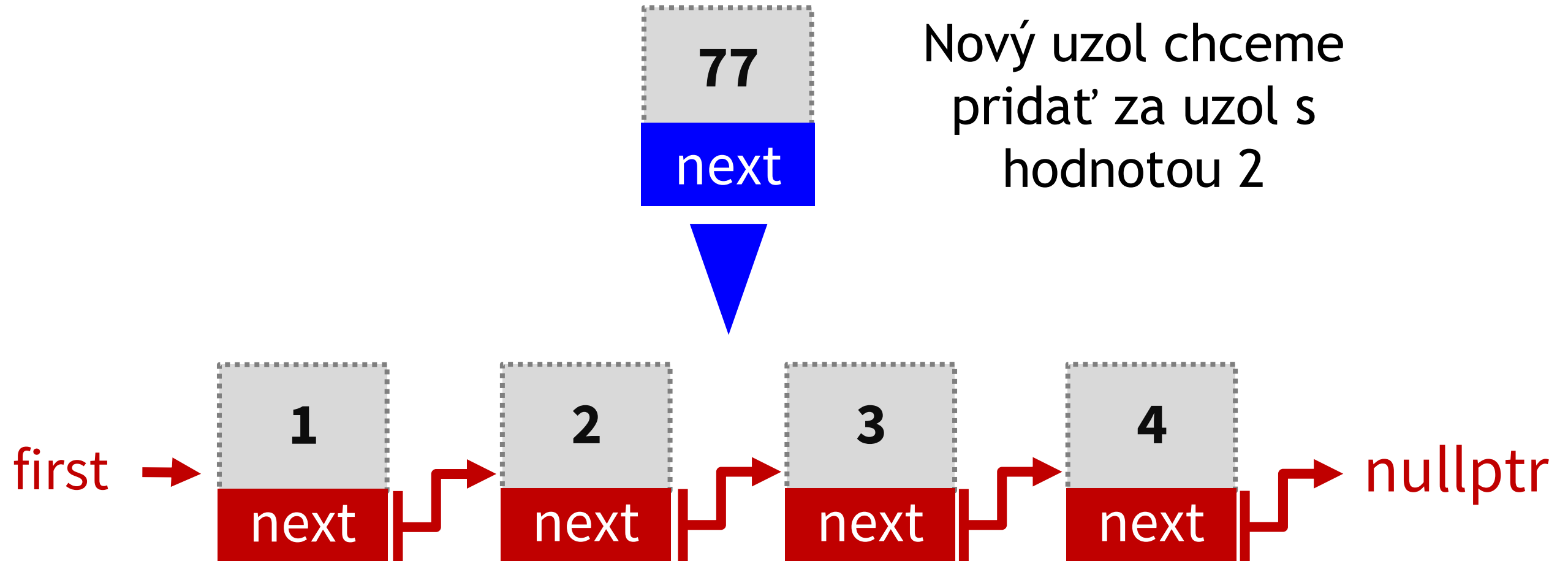
Vymazanie prvého uzla

Vymazanie i-teho uzla

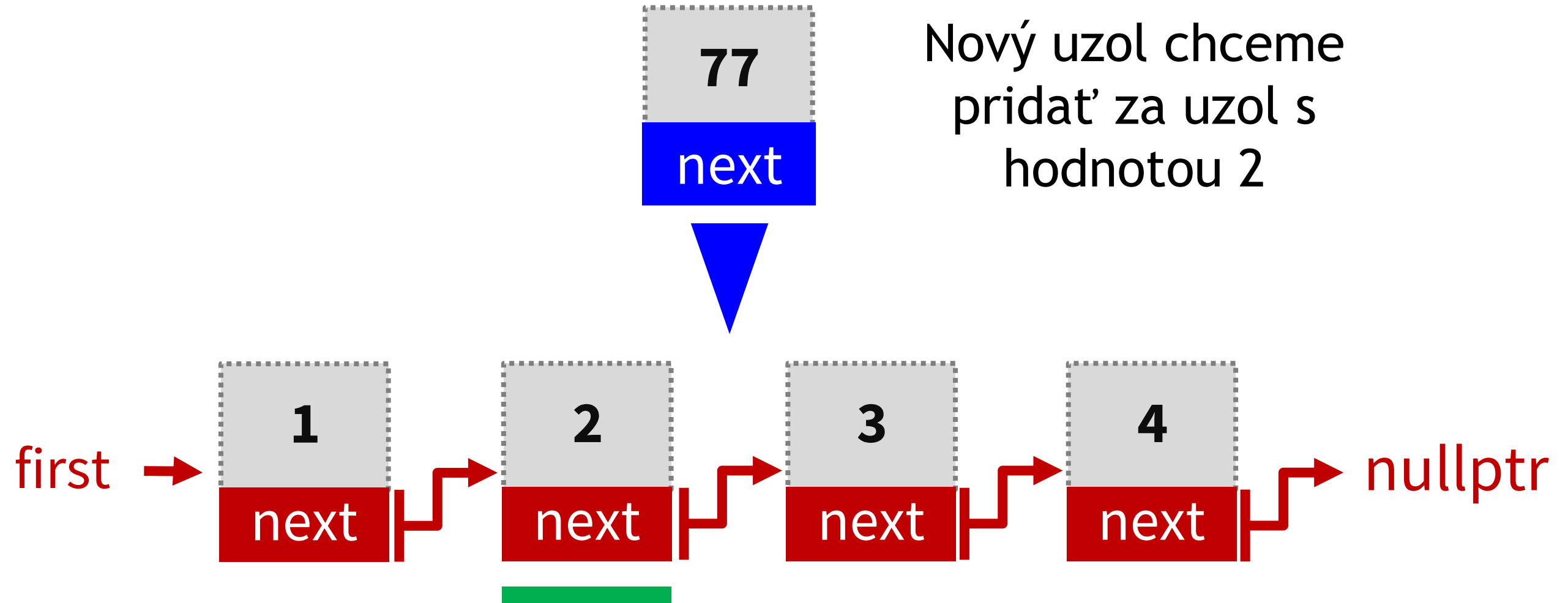
Pridanie uzla do stredy



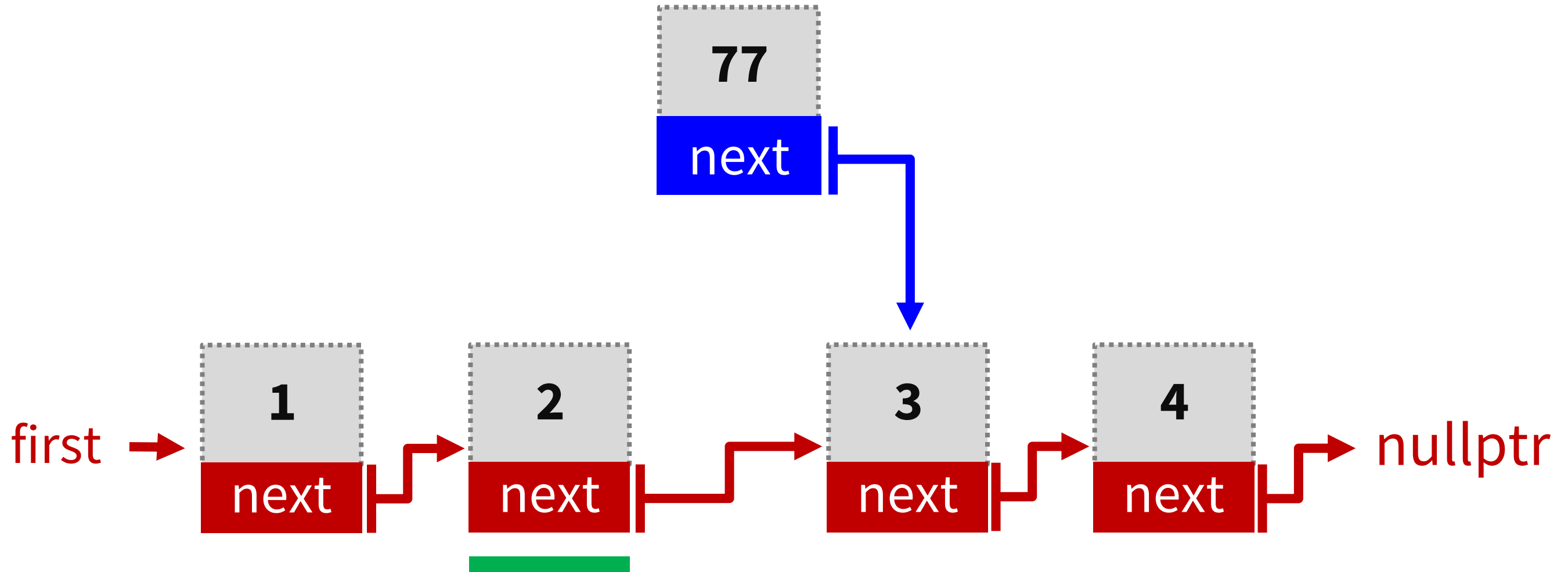
Pridanie uzla do stredu



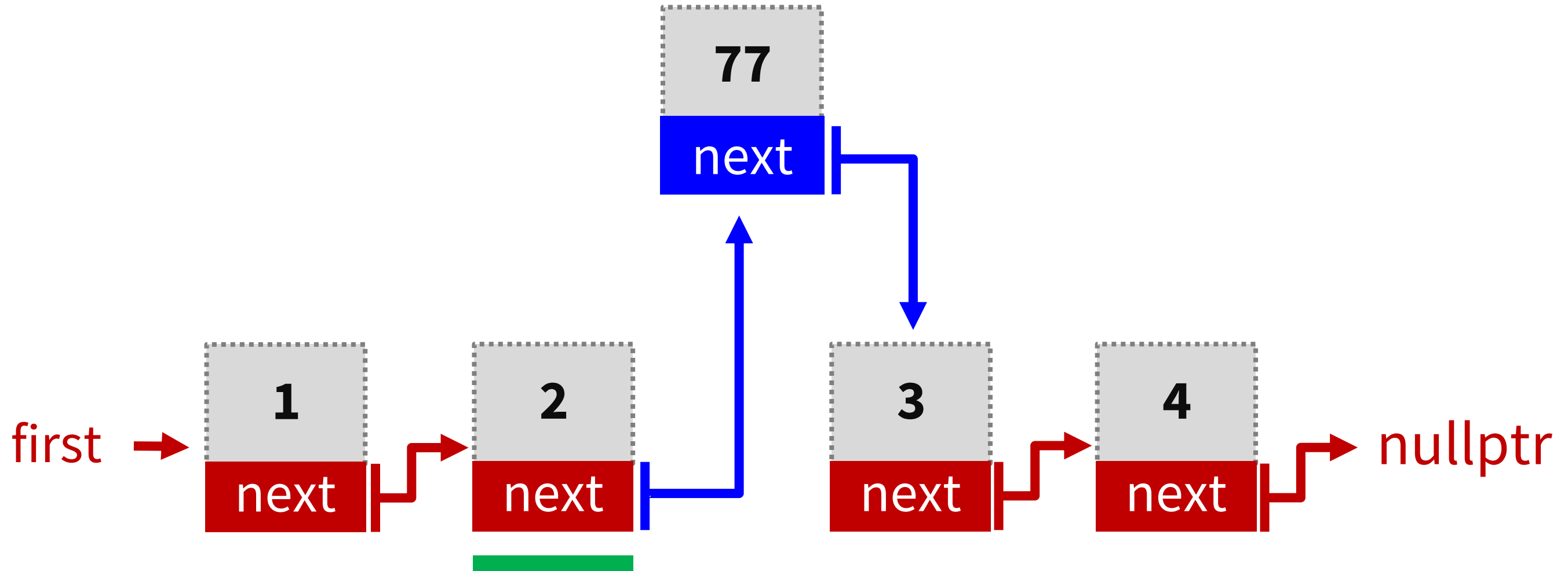
Pridanie uzla do streda



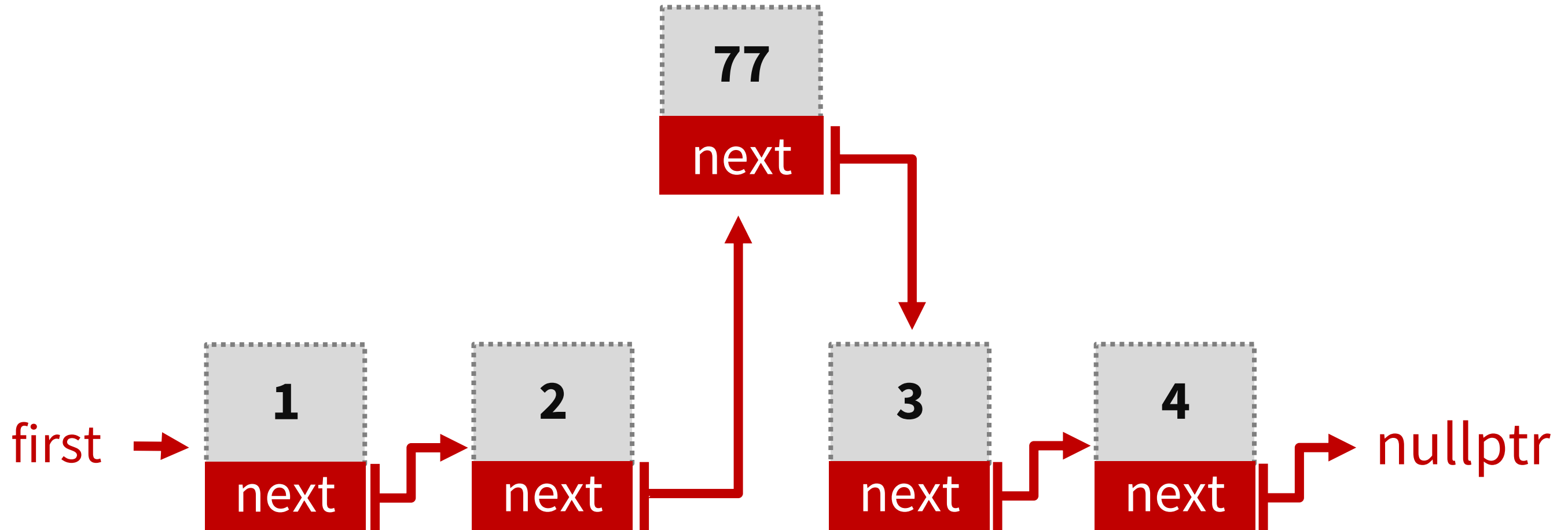
Pridanie uzla do stredy



Pridanie uzla do stredu



Pridanie uzla do stredu



Operácie:

Pridanie uzla na začiatok

Pridanie uzla do stredu

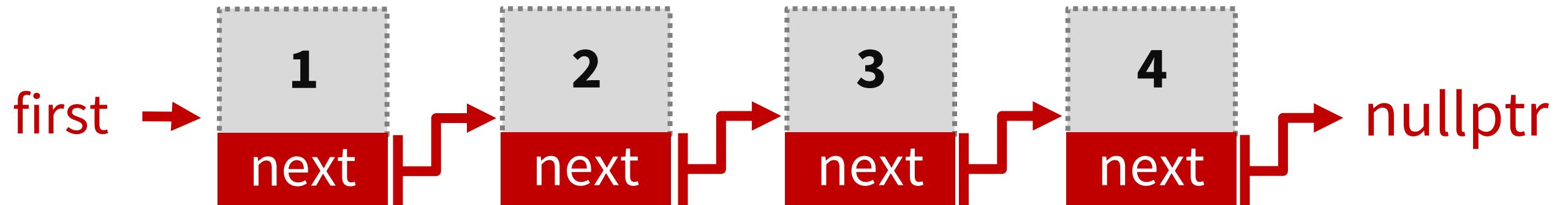
Pridanie uzla na koniec •

Výpis zoznamu

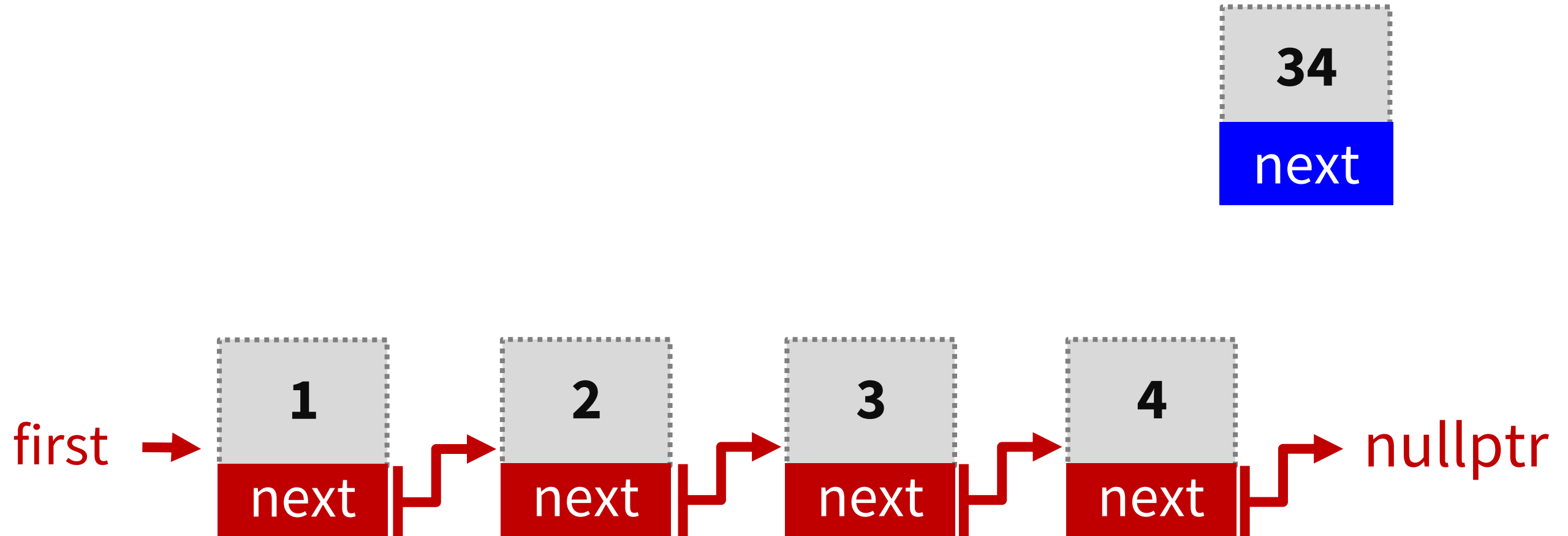
Vymazanie prvého uzla

Vymazanie i-teho uzla

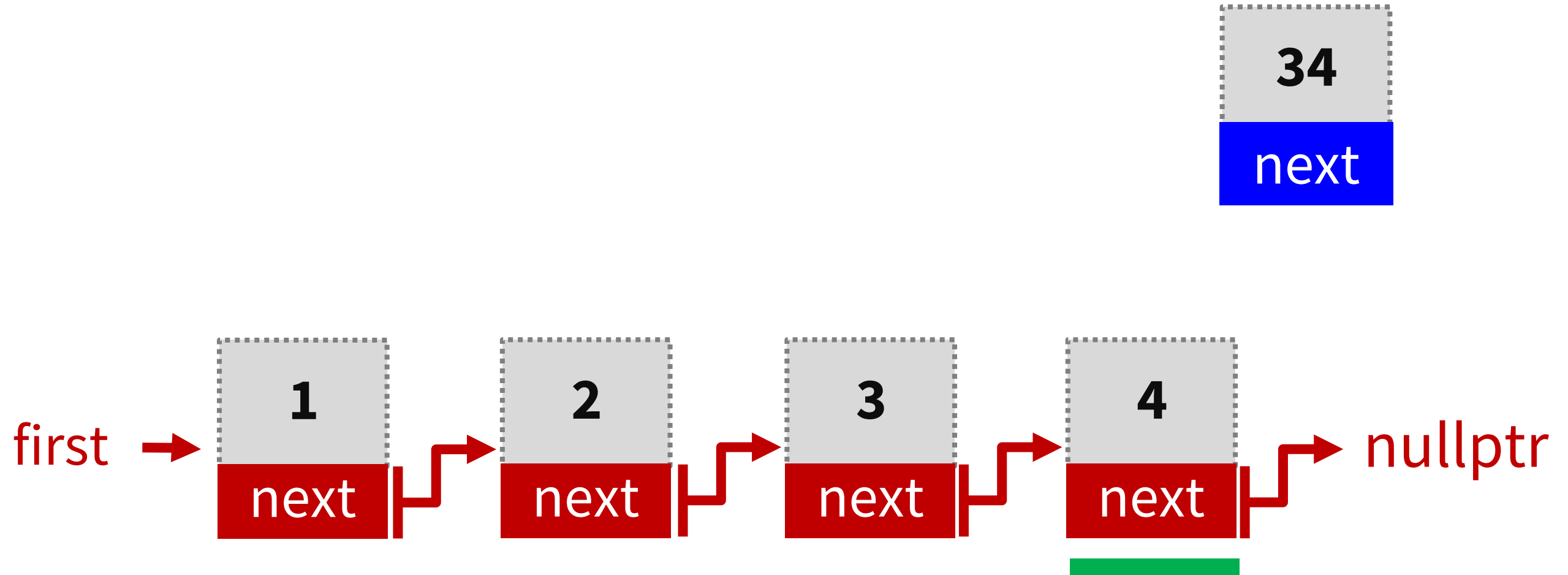
Pridanie uzla na koniec



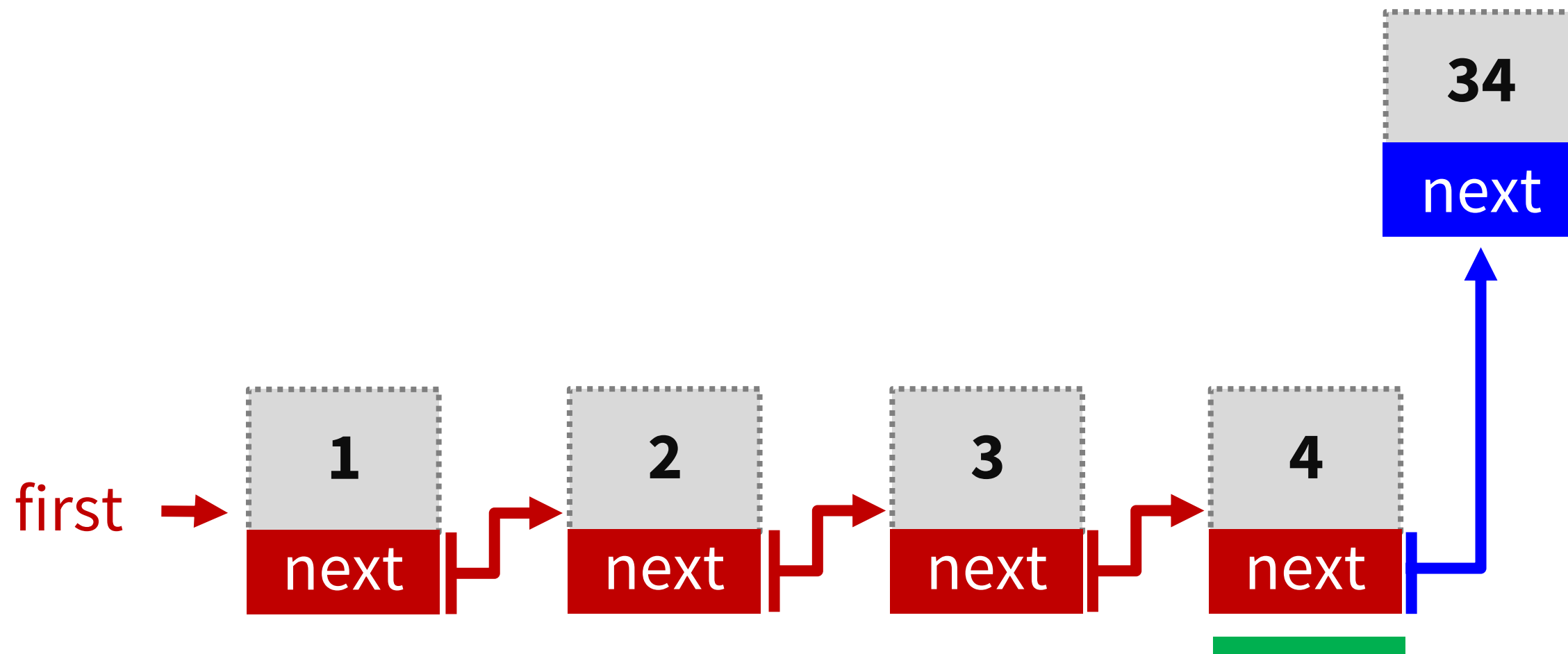
Pridanie uzla na koniec



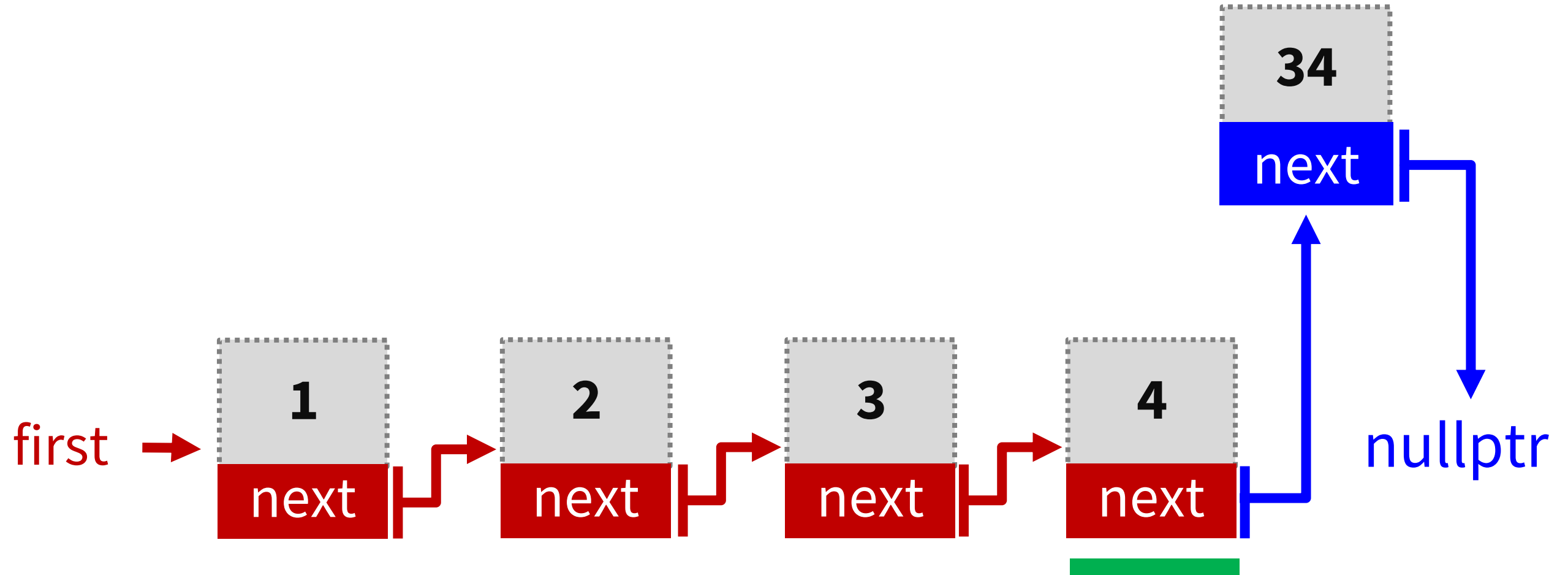
Pridanie uzla na koniec



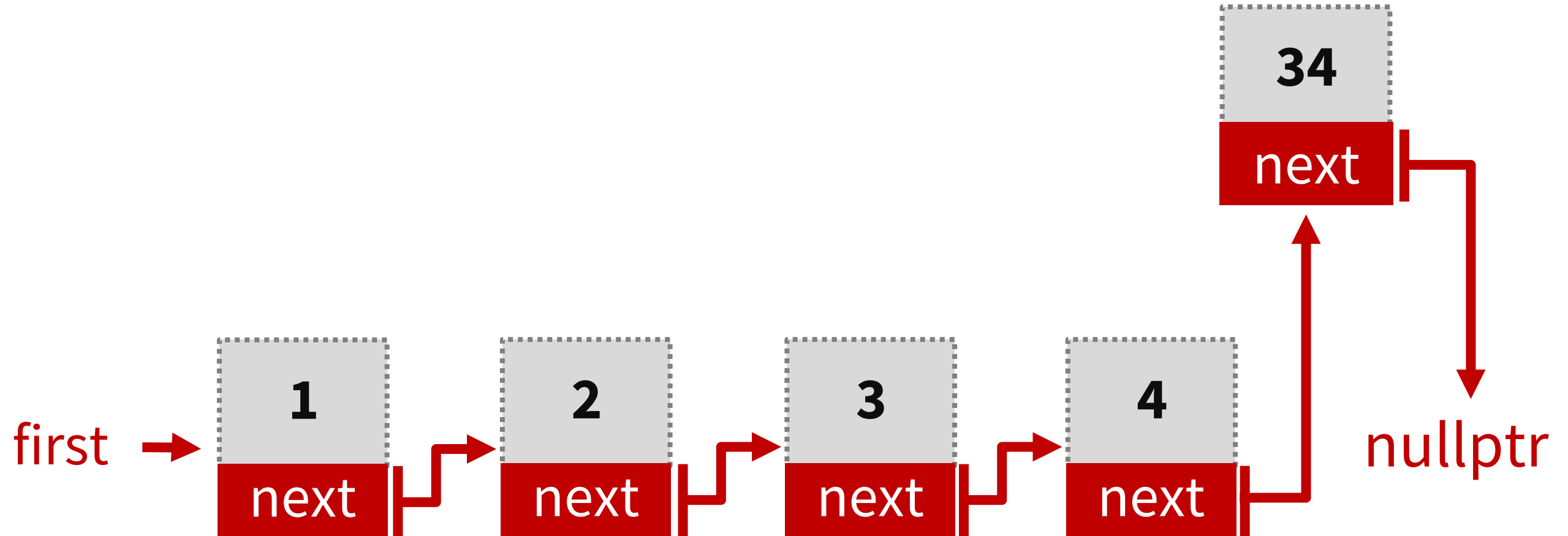
Pridanie uzla na koniec



Pridanie uzla na koniec



Pridanie uzla na koniec



Operácie:

Pridanie uzla na začiatok

Pridanie uzla do stredu

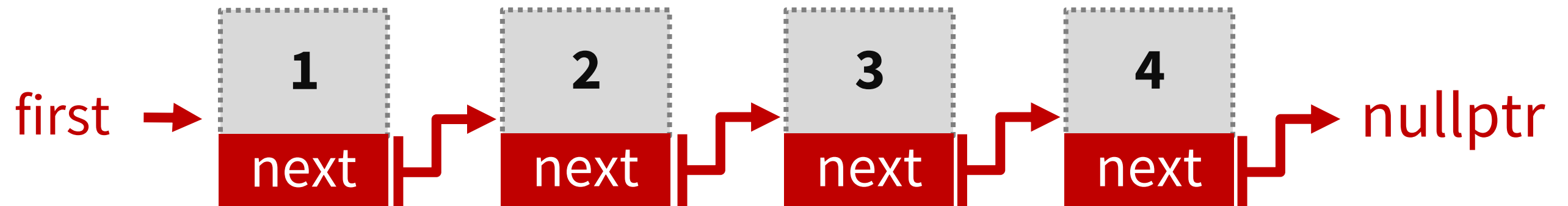
Pridanie uzla na koniec

Prechod zoznamom •

Vymazanie prvého uzla

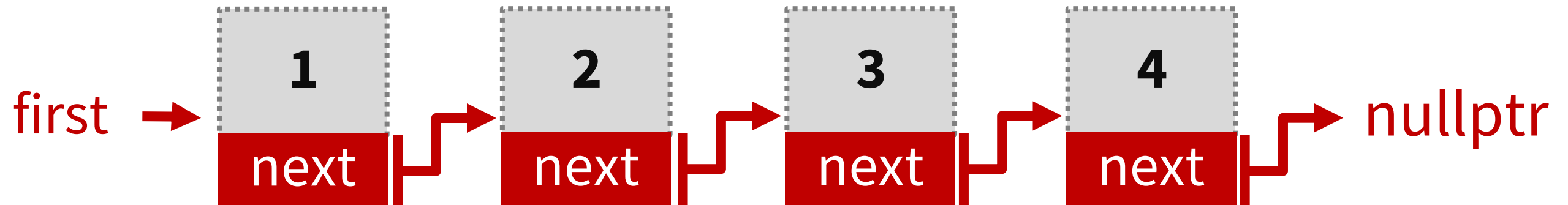
Vymazanie i-teho uzla

Prechod zoznamom



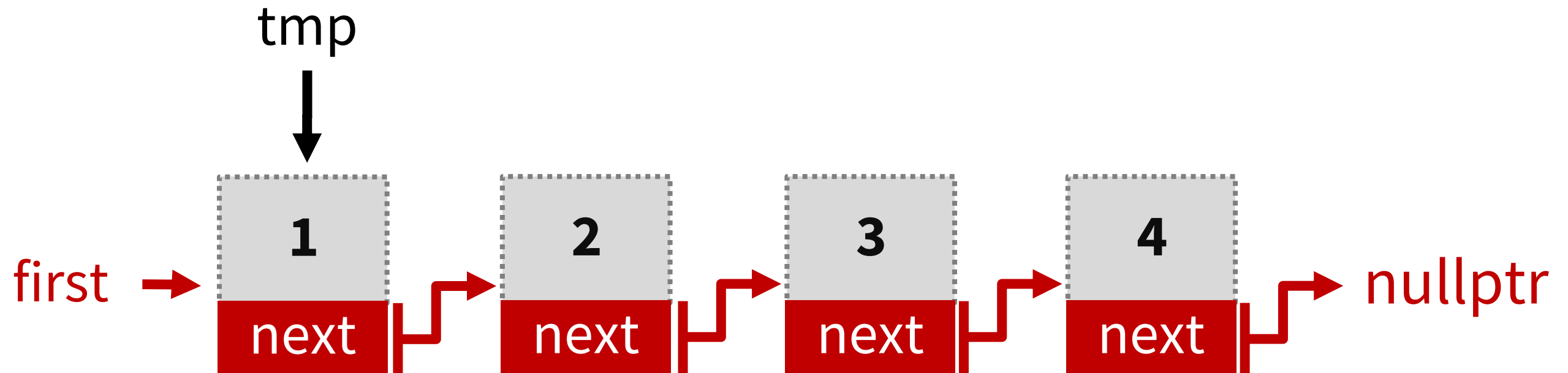
Prechod zoznamom

```
Node* tmp = list->first;
```

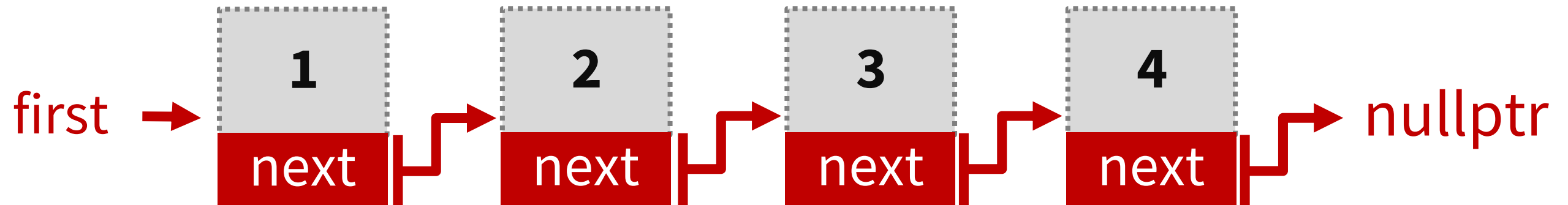


Prechod zoznamom

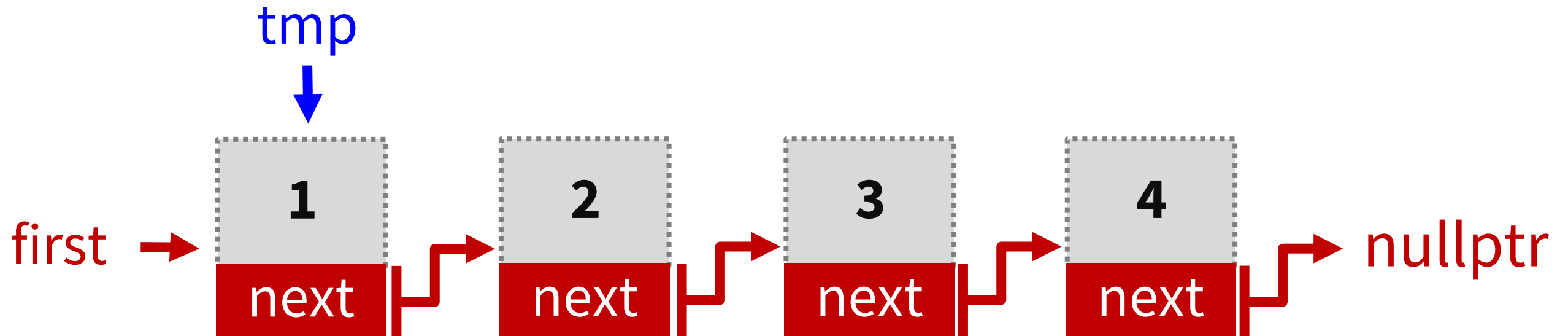
```
Node* tmp = list->first;
```



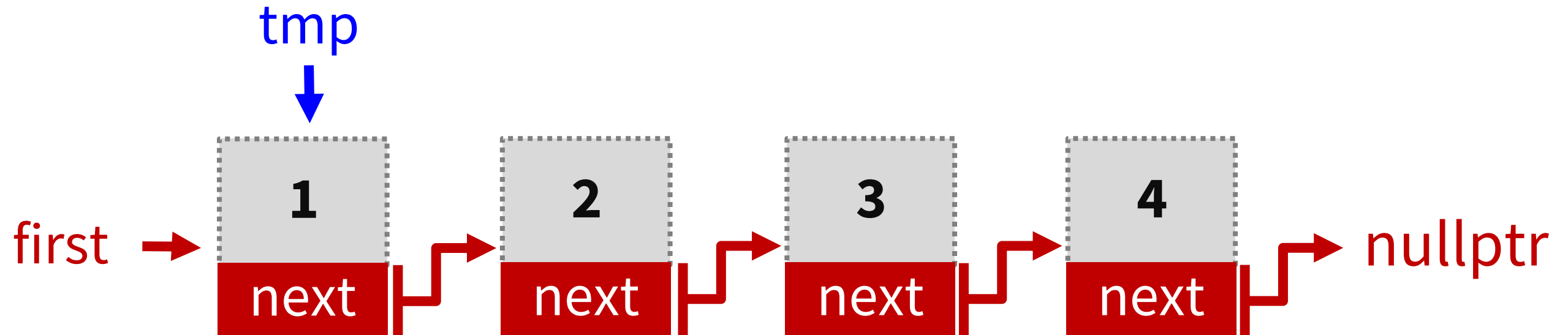
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



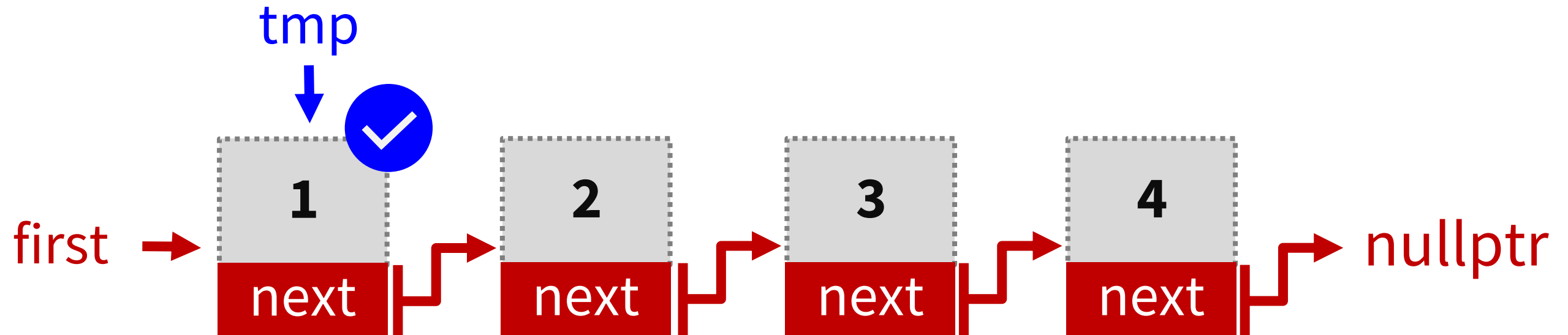
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



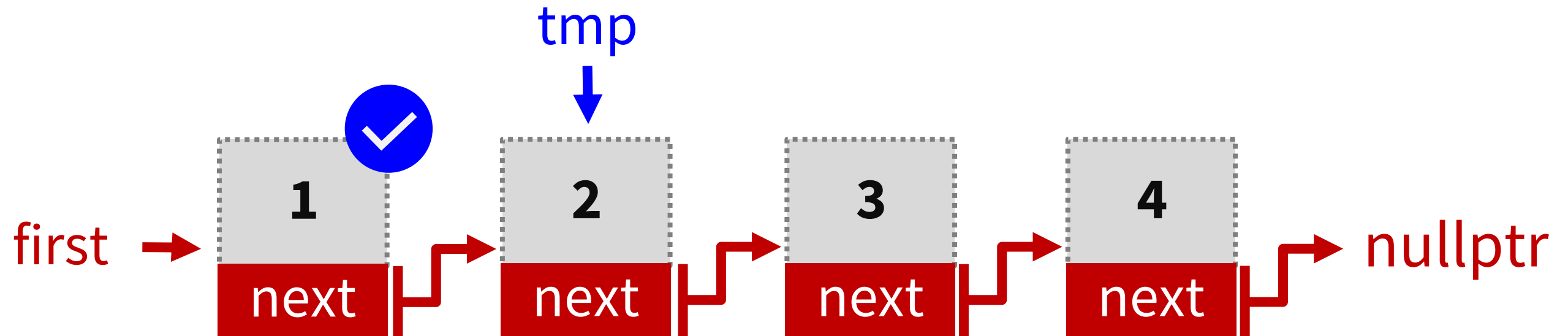
```
void printList(const List * list){  
    Node* tmp = list->first;  
    OK while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



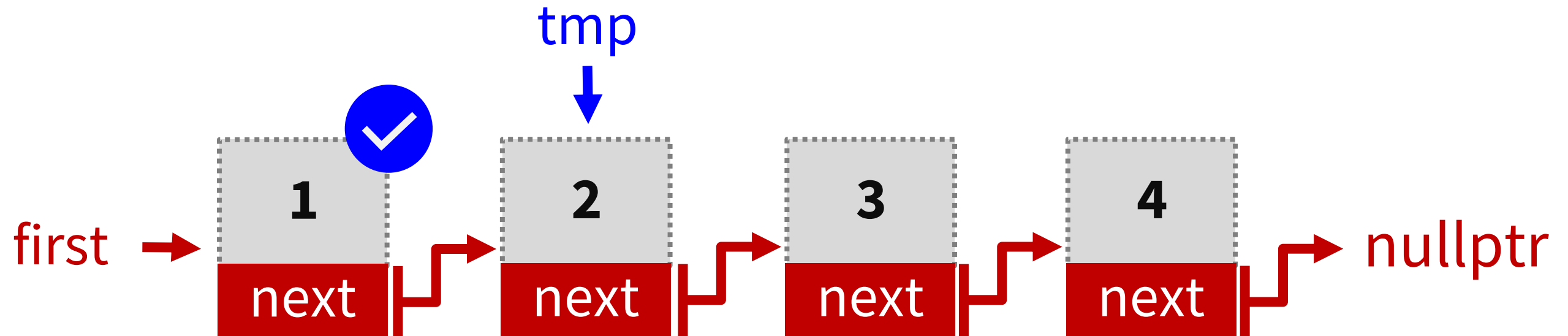

```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



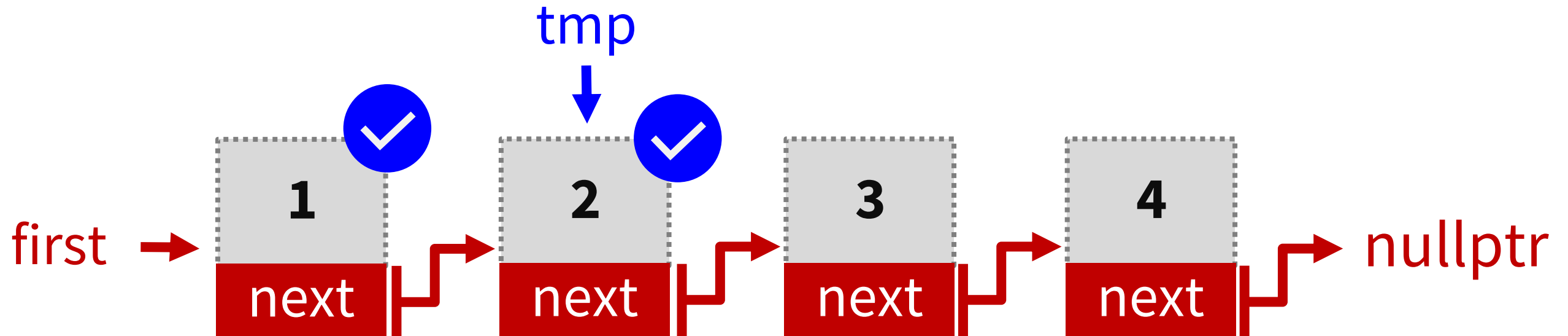
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



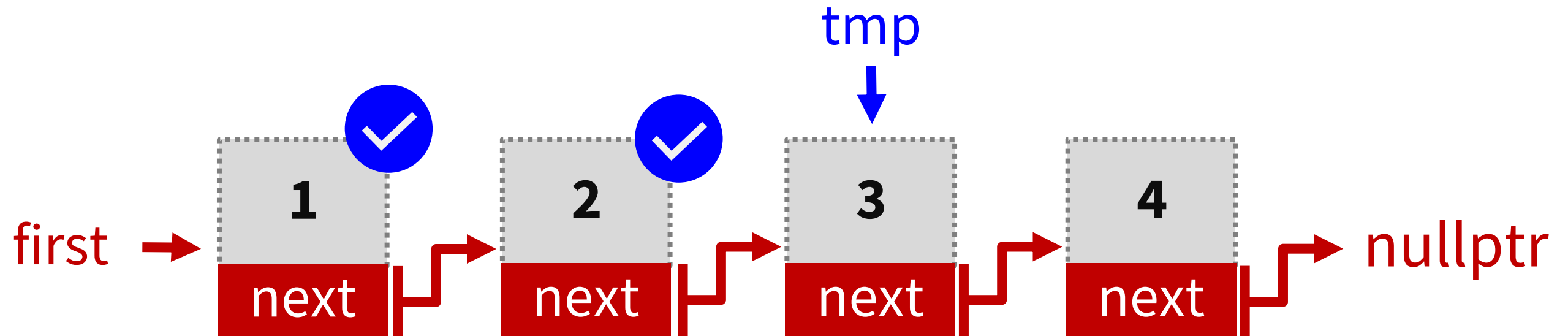
```
void printList(const List * list){  
    Node* tmp = list->first;  
    OK while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



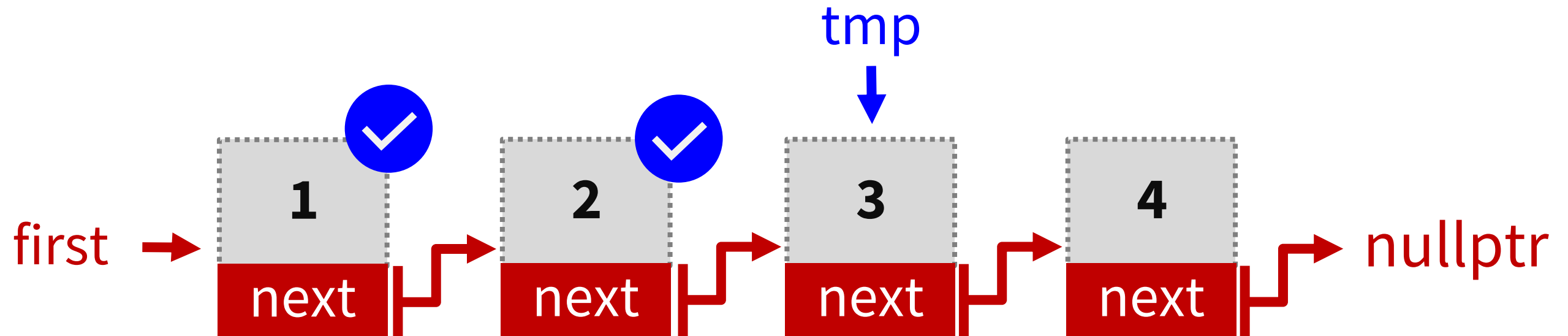
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



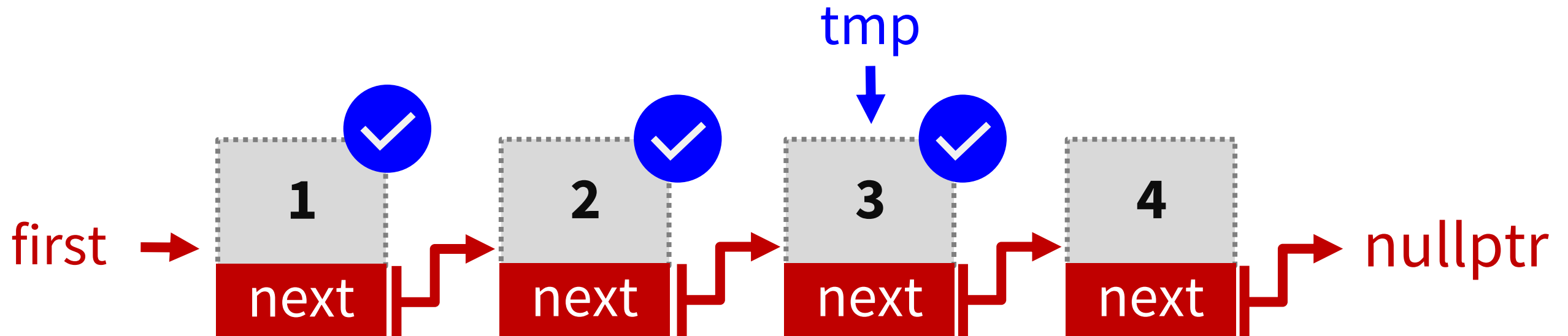
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



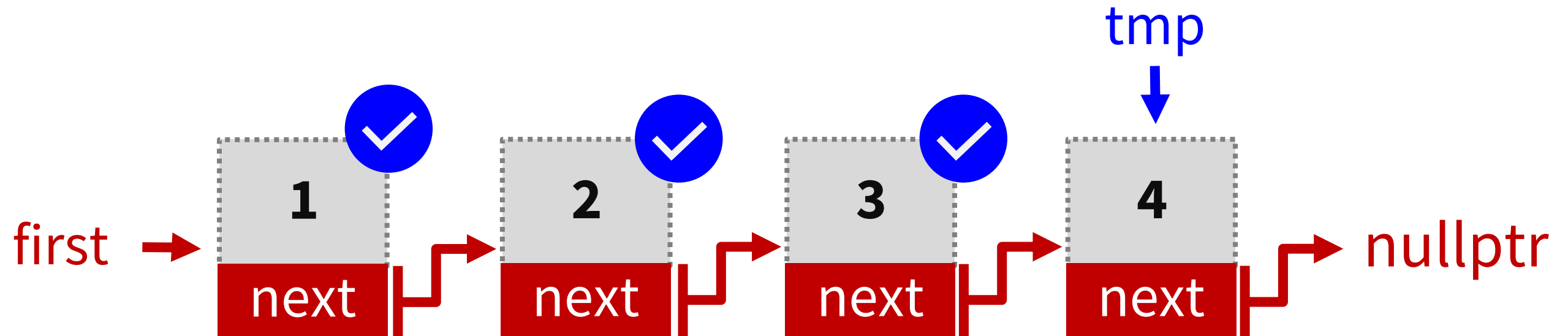
```
void printList(const List * list){  
    Node* tmp = list->first;  
    OK while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



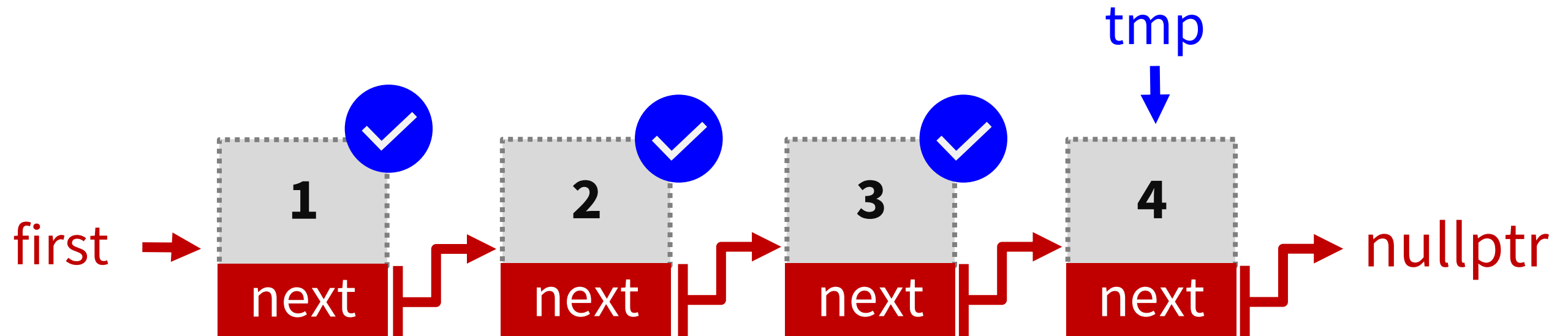
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



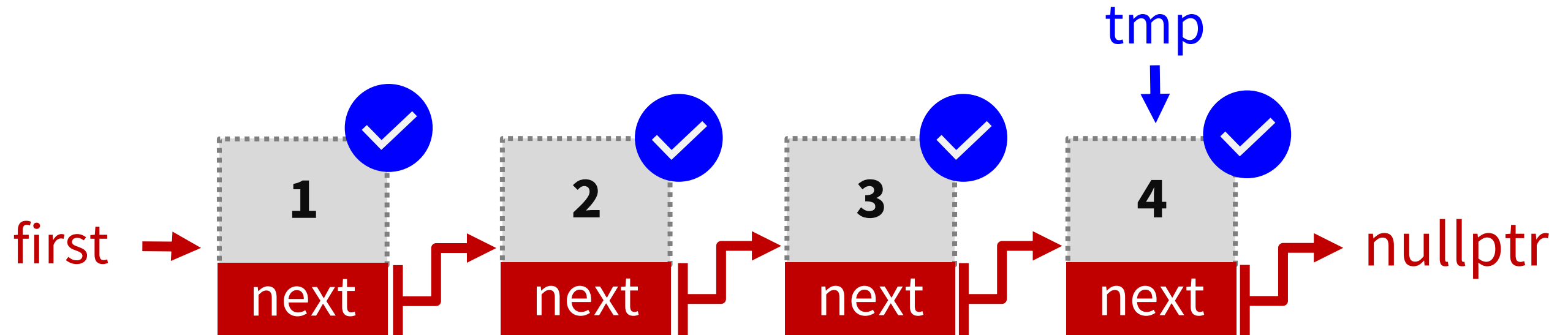
```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



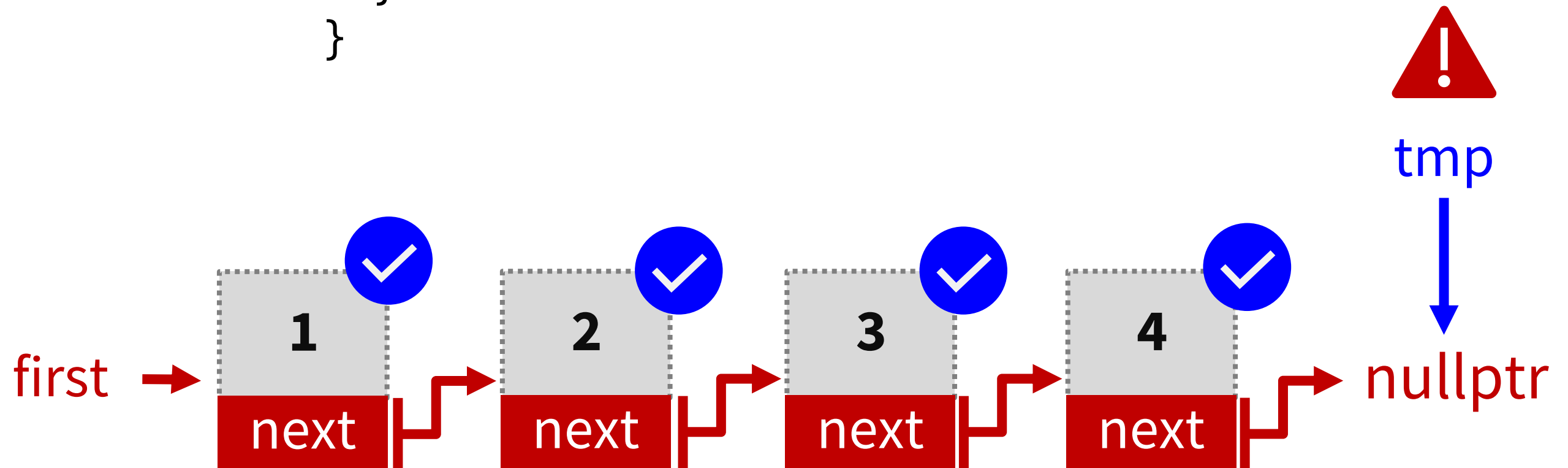

```
void printList(const List * list){  
    Node* tmp = list->first;  
    OK while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```

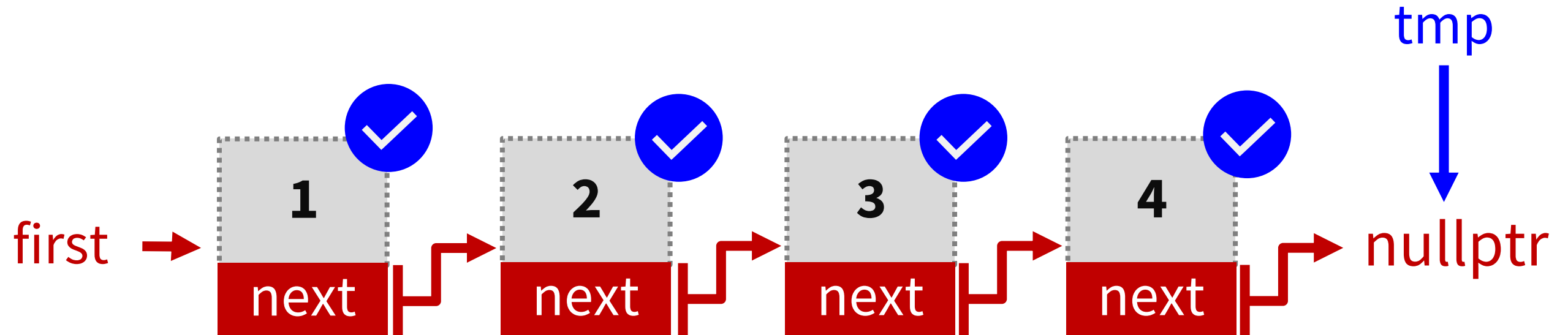


```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```





```
void printList(const List * list){  
    Node* tmp = list->first;  
    while(tmp){  
        std::cout << tmp->data << " ";  
        tmp = tmp->next;  
    }  
}
```



Operácie:

Pridanie uzla na začiatok

Pridanie uzla do stredu

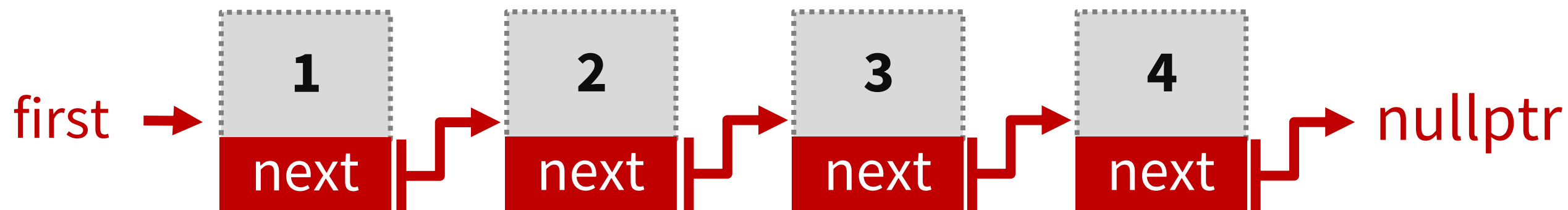
Pridanie uzla na koniec

Výpis zoznamu

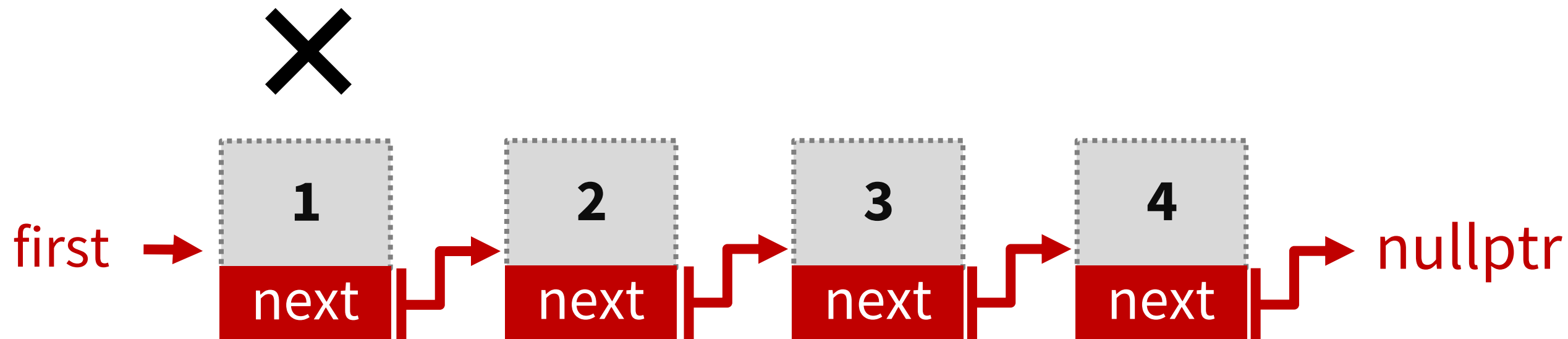
Vymazanie prvého uzla •

Vymazanie i-teho uzla

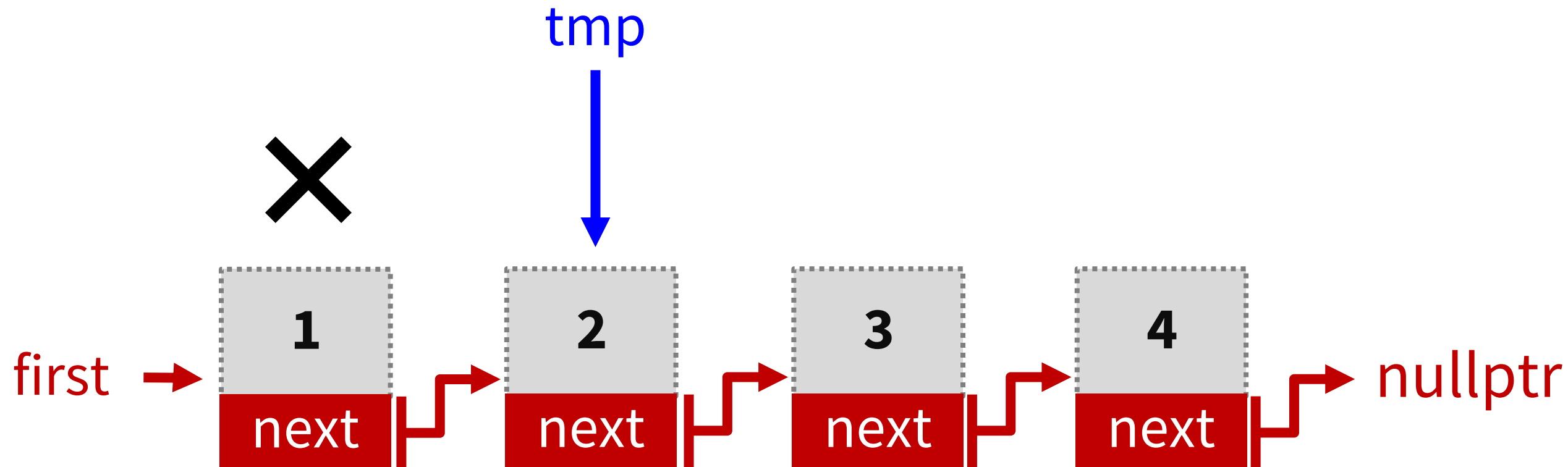
Vymazanie prvého uzla



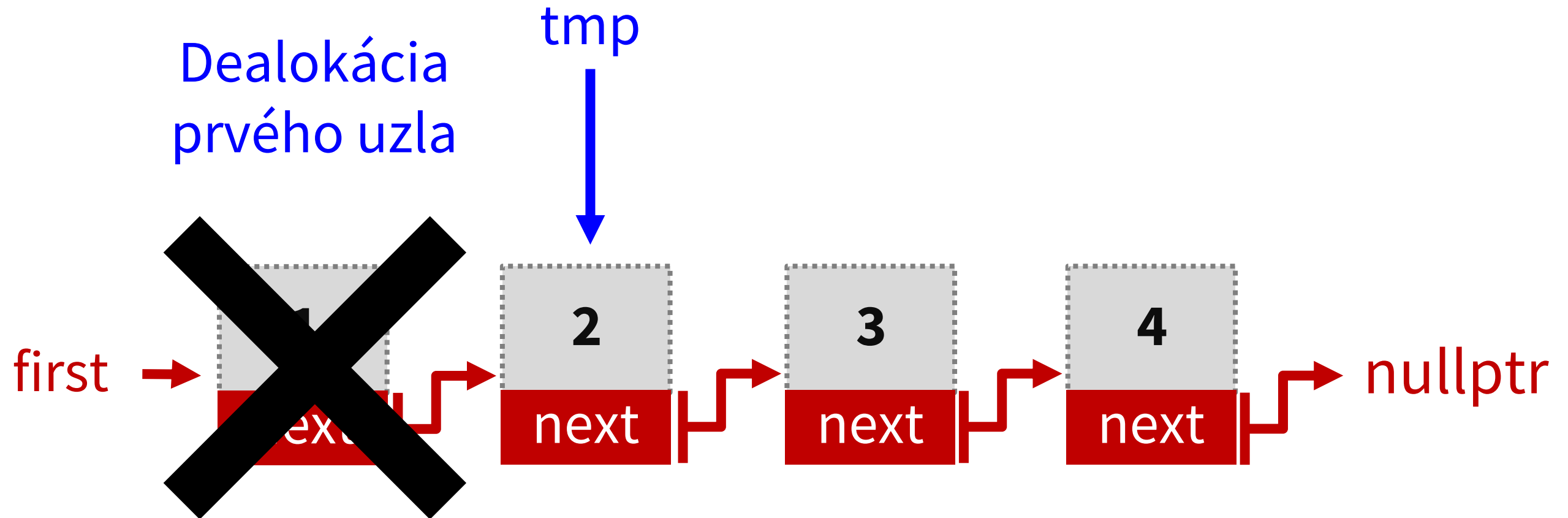
Vymazanie prvého uzla



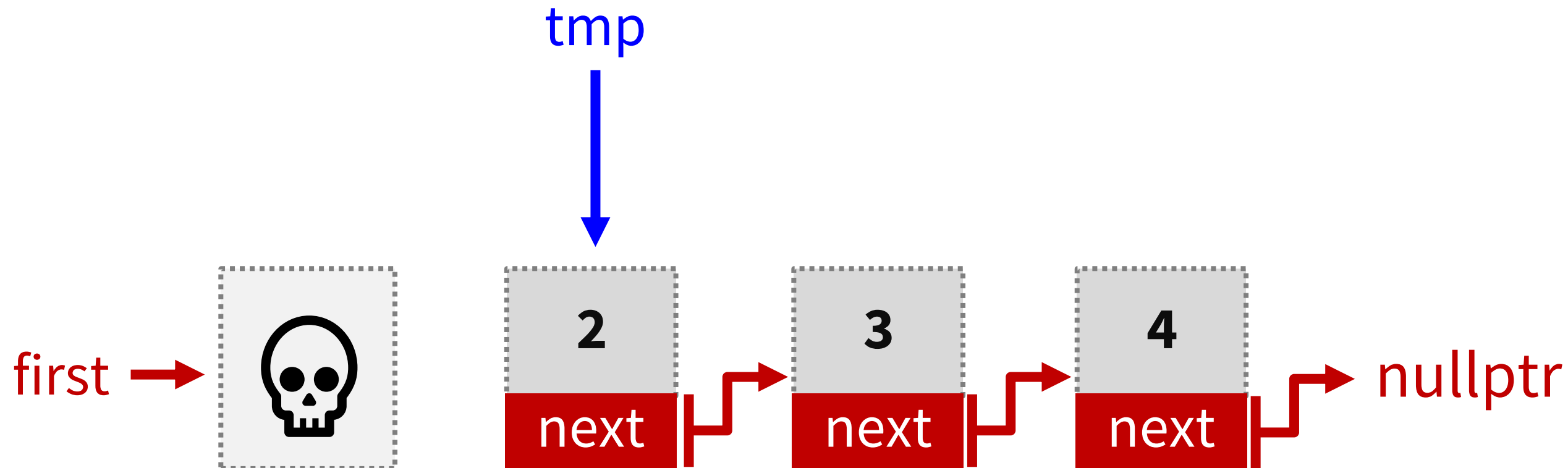
Vymazanie prvého uzla



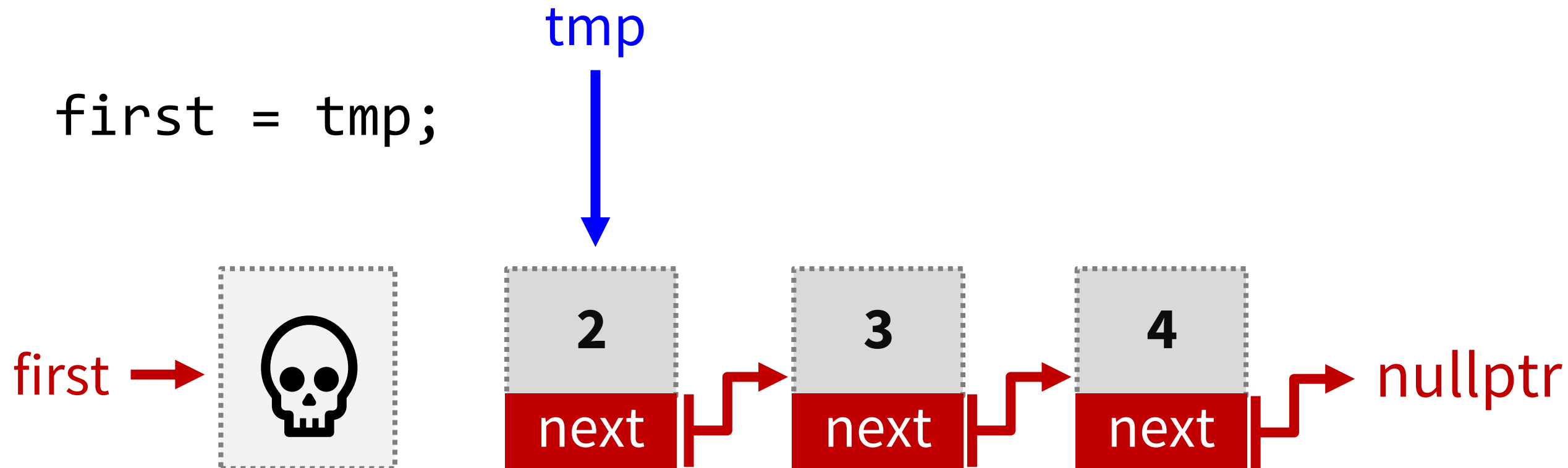
Vymazanie prvého uzla



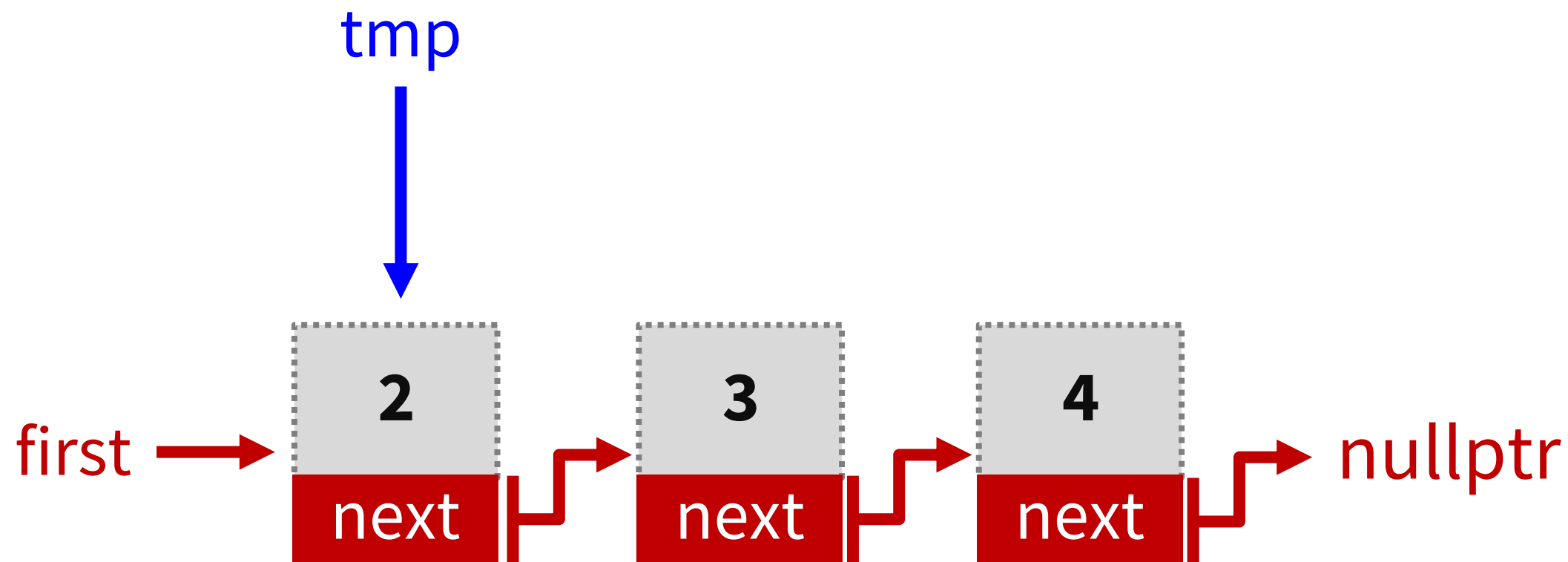
Vymazanie prvého uzla



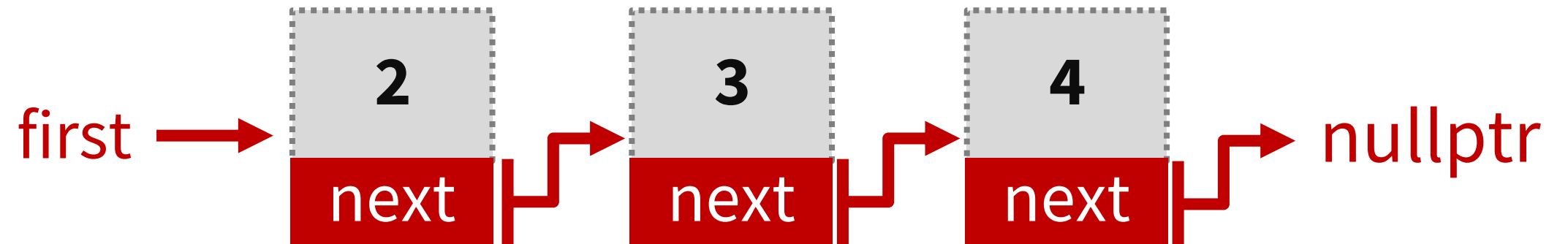
Vymazanie prvého uzla



Vymazanie prvého uzla



Vymazanie prvého uzla



Operácie:

Pridanie uzla na začiatok

Pridanie uzla do stredu

Pridanie uzla na koniec

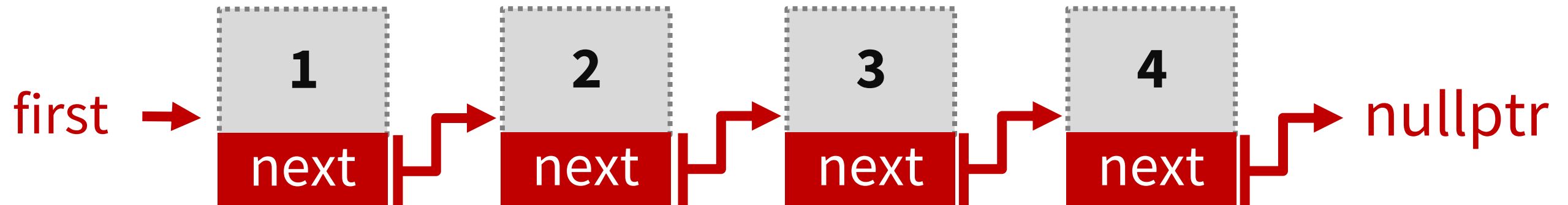
Výpis zoznamu

Vymazanie prvého uzla

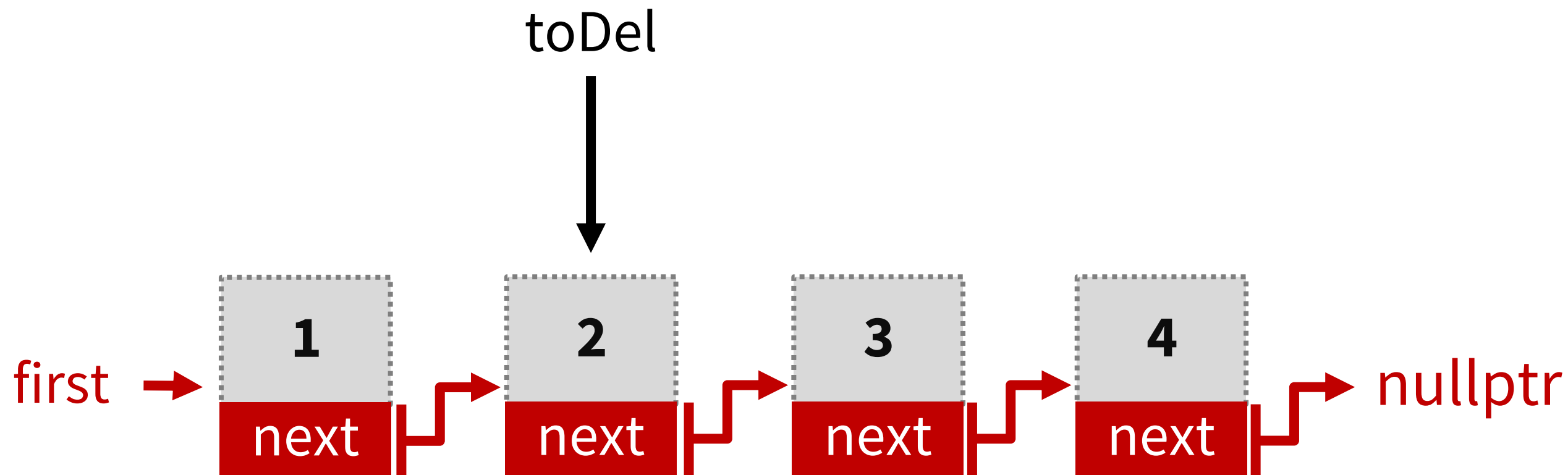
Vymazanie i-teho uzla •

Vymazanie i-teho uzla

×

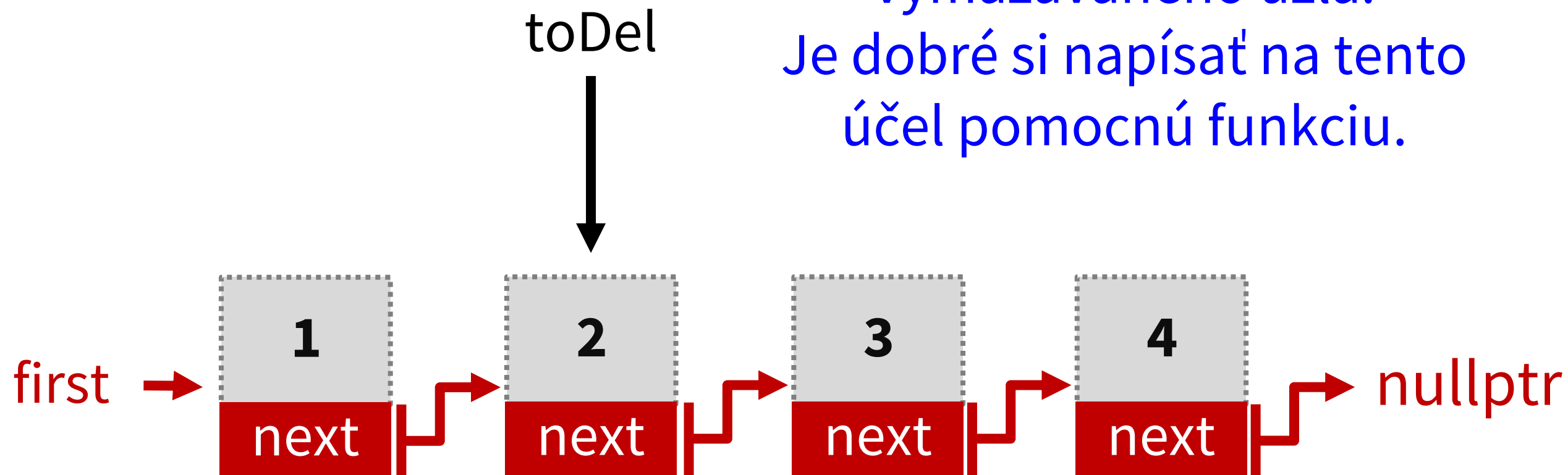


Vymazanie i-teho uzla

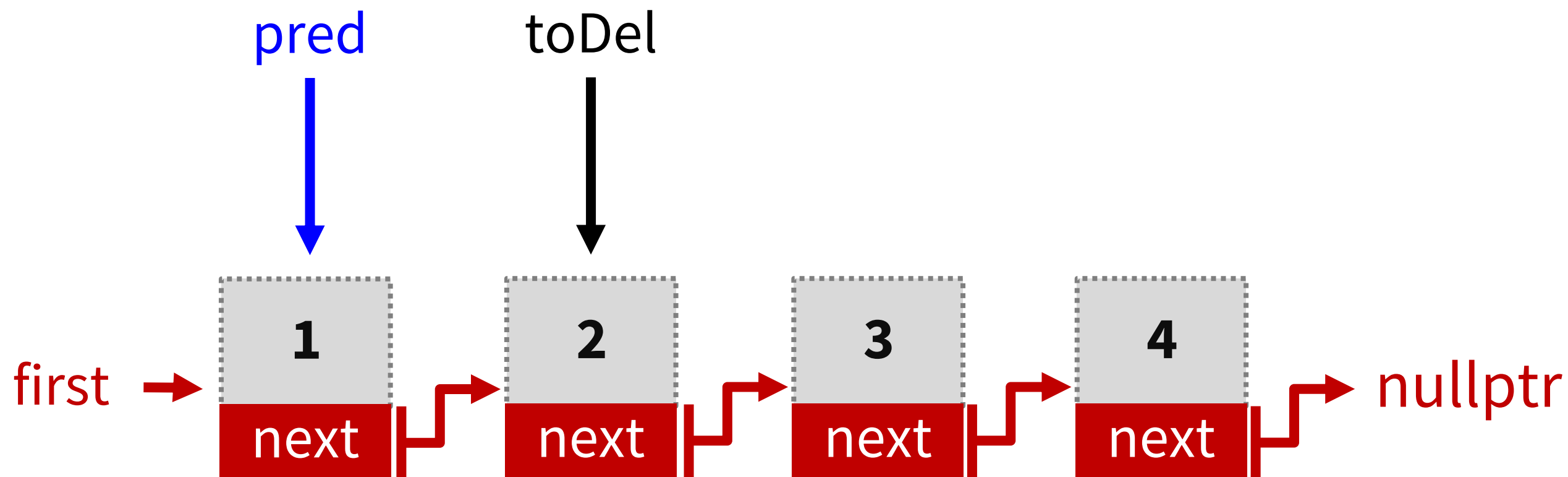


Vymazanie i-teho uzla

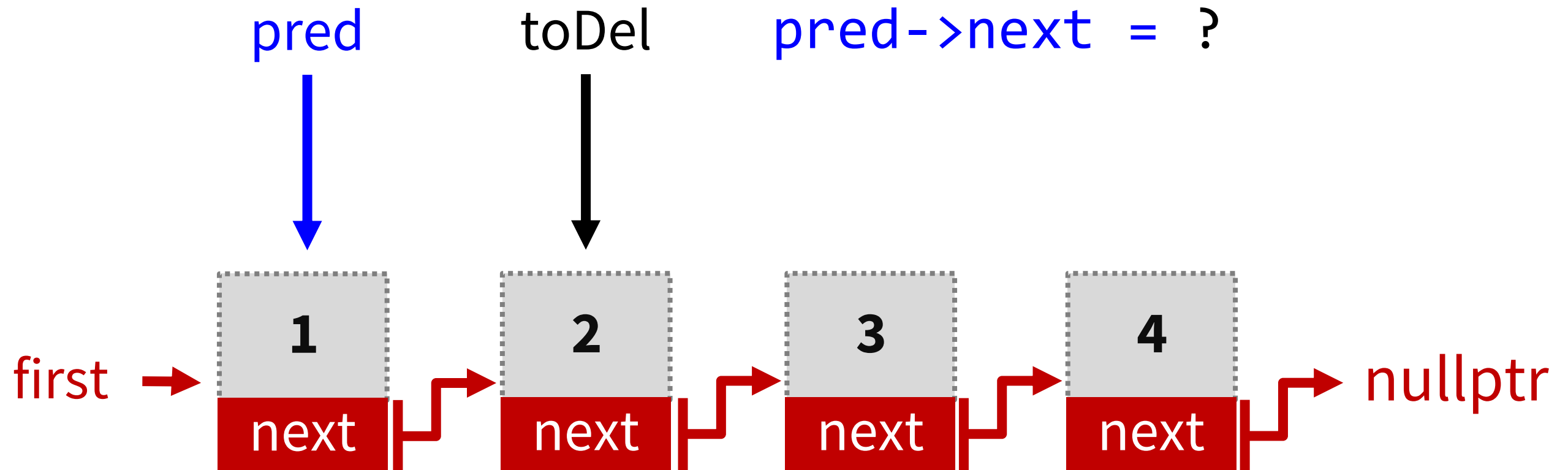
Musíme nájsť predchodcu
vymazávaného uzla.
Je dobré si napísať na tento
účel pomocnú funkciu.



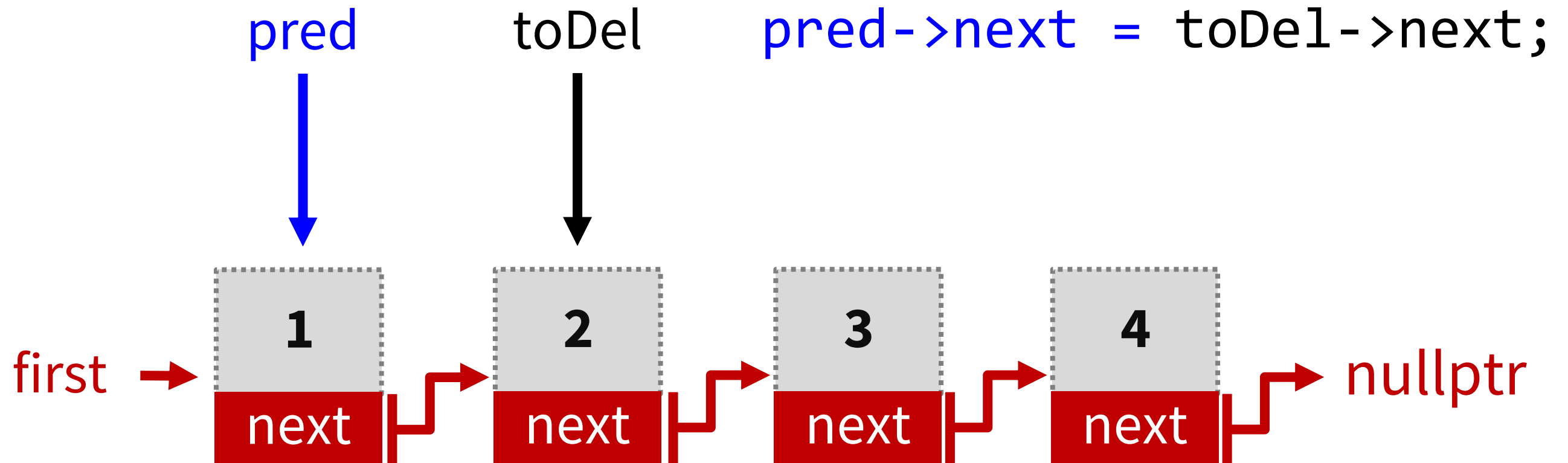
Vymazanie i-teho uzla



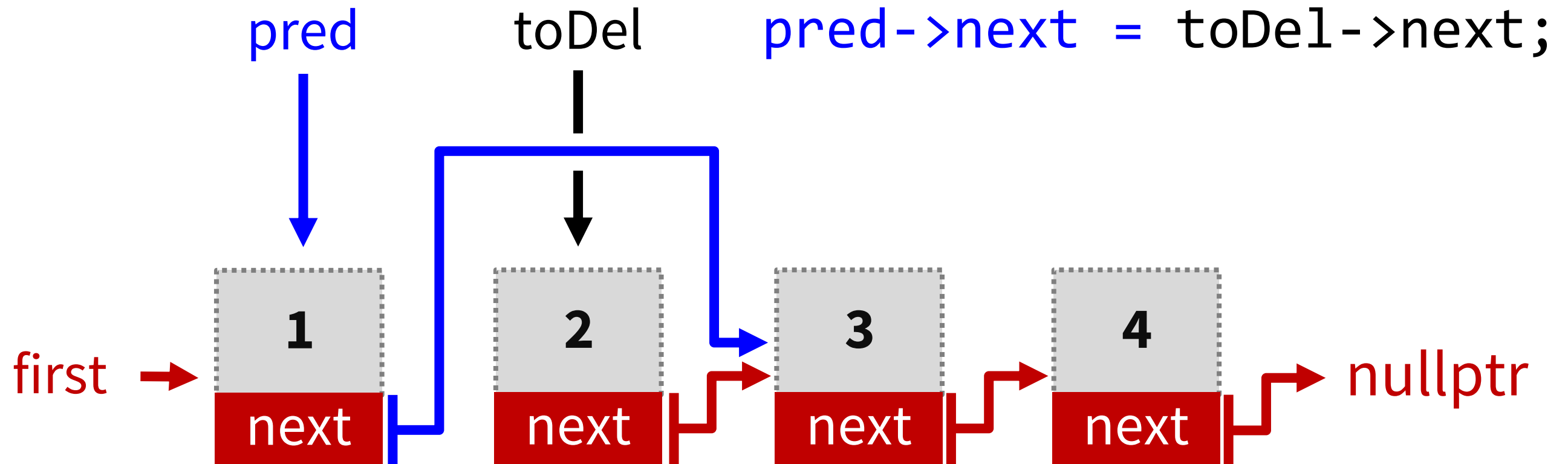
Vymazanie i-teho uzla



Vymazanie i-teho uzla

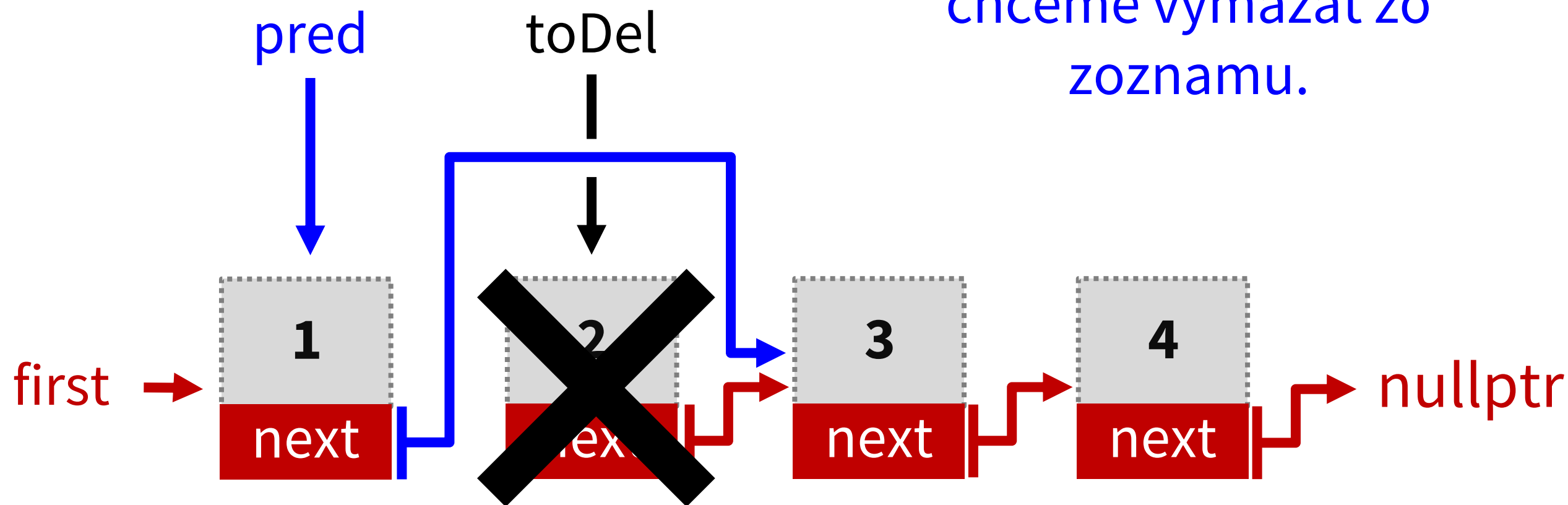


Vymazanie i-teho uzla



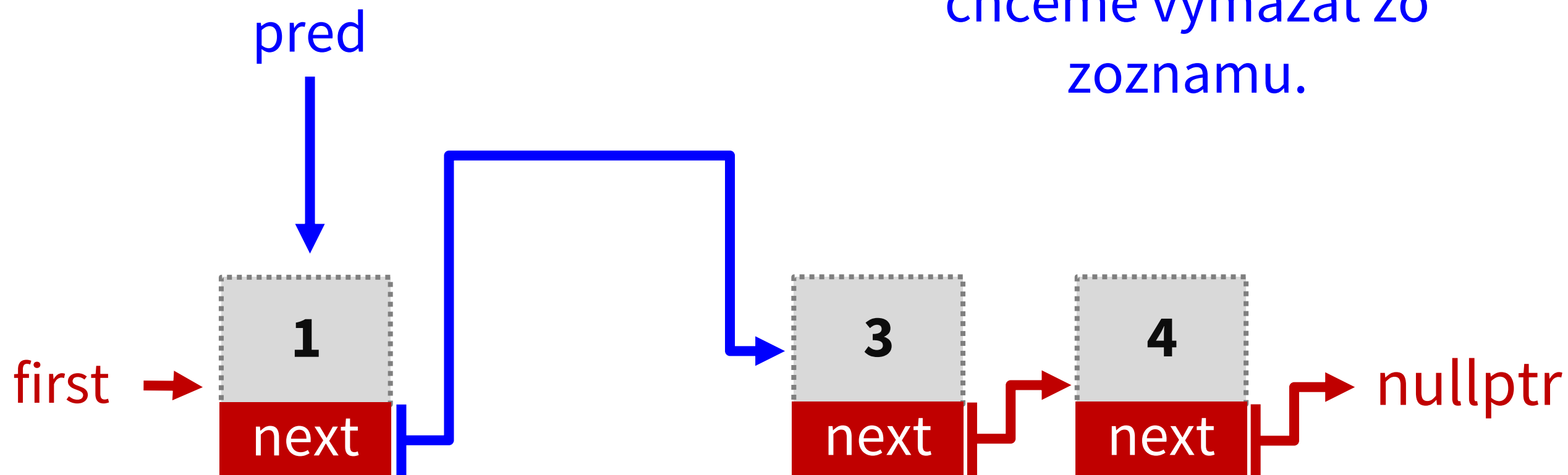
Vymazanie i-teho uzla

Dealokujeme uzol, ktorý
chceme vymazať zo
zoznamu.



Vymazanie i-teho uzla

Dealokujeme uzol, ktorý
chceme vymazať zo
zoznamu.



Vymazanie i-teho uzla

