Algoritmy
cvičení

Spojový seznam

- Datová struktura umožňující uchovat velké množství hodnot
- Obsahuje položky stejného typu
- Položky jsou navzájem provázány vzájemnými odkazy pomocí ukazatelů
  - Místo jednoho velkého souvislého úseku paměti (který používají pole) budeme mít paměť pro každý prvek zvlášť, tyto části paměti propojíme odkazem

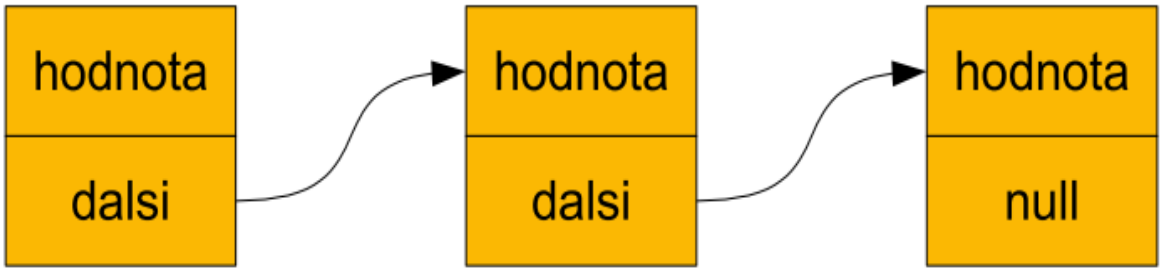
Spojový seznam

- Datová struktura umožňující uchovat velké množství hodnot
- Obsahuje položky stejného typu
- Položky jsou navzájem provázány vzájemnými odkazy pomocí ukazatelů
  - Místo jednoho velkého souvislého úseku paměti (který používají pole) budeme mít paměť pro každý prvek zvlášť, tyto části paměti propojíme odkazem
- Zřetězený seznam je buď prázdný prvek nebo obsahuje hodnotu a připojený zřetězený seznam

Spojový seznam

```
class Prvek
{
    public int hodnota;
    public Prvek dalsi;
}
```

```
typedef struct Prvek{
    int hodnota;
    Prvek* dalsi;
};
```

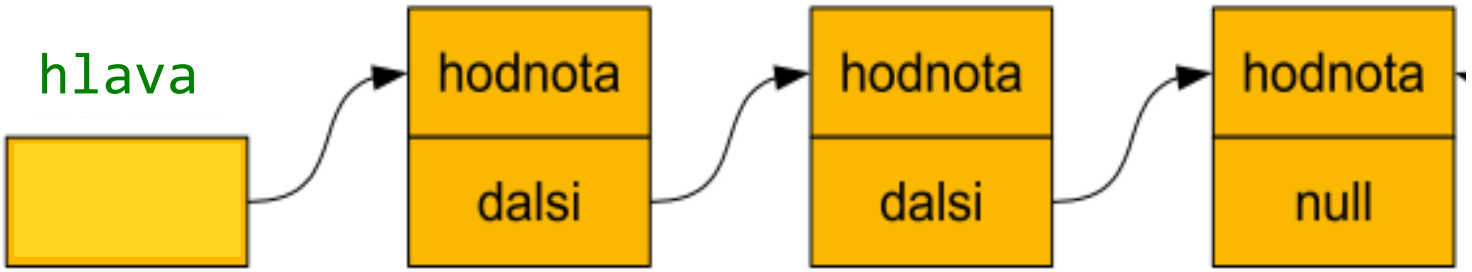


Spojový seznam

- Na začátek seznamu ukazuje hlavička
- Konec se pozná tak, že neexistuje další prvek

```
class Seznam
{
    public Prvek hlava;
    public Seznam(int cislo) {
        Prvek p=new Prvek();
        p.hodnota=cislo;
        p.dalsi=null;
        hlava = p;
    }
}
```

```
class Seznam{
    Prvek* hlava;
    Seznam(int cislo){
        Prvek p;
        p.hodnota=cislo;
        p.dalsi=NULL;
        hlava=&p;
    }
}
```



Příklad

- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

Příklad

- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

```
class Prvek
{
    public int hodnota;
    public Prvek dalsi;
}
class Seznam
{
    public Prvek hlava;
    public Seznam(int cislo) {
        Prvek p=new Prvek();
        p.hodnota=cislo;
        p.dalsi=null;
        hlava = p;
    }
}
```

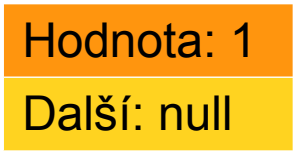
```
typedef struct Prvek{
    int hodnota;
    Prvek* dalsi;
};

class Seznam{
    Prvek* hlava;
    Seznam(int cislo){
        Prvek p;
        p.hodnota=cislo;
        p.dalsi=NULL;
        hlava=&p;
    }
};
```

Příklad

- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

```
public void vytvor123(){
   Prvek jedna, dva, tri;
   jedna=new Prvek(); jedna.hodnota=1;
}
```




```
void vytvor123(){
    Prvek jedna, dva, tri;
    jedna.hodnota = 1;
}
```

Příklad

- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

```
public void vytvor123(){
    Prvek jedna, dva, tri;
    jedna=new Prvek(); jedna.hodnota=1;
    dva=new Prvek(); dva.hodnota=2;
    tri=new Prvek(); tri.hodnota=3;
}
```

Hodnota: 1
Další: null

```
void vytvor123(){
    Prvek jedna, dva, tri;
    jedna.hodnota = 1;
    dva.hodnota = 2;
    tri.hodnota = 3;;
}
```

Hodnota: 2
Další: null

Hodnota: 3
Další: null

Příklad

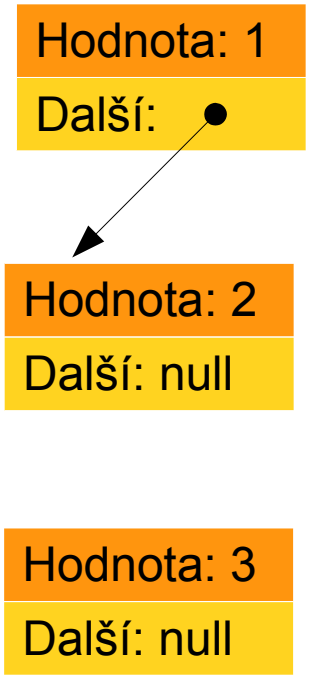
- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

```
public void vytvor123(){
    Prvek jedna, dva, tri;
    jedna=new Prvek(); jedna.hodnota=1;
    dva=new Prvek(); dva.hodnota=2;
    tri=new Prvek(); tri.hodnota=3;

    jedna.dalsi=dva;
}
```

```
void vytvor123(){
    Prvek jedna, dva, tri;
    jedna.hodnota = 1;
    dva.hodnota = 2;
    tri.hodnota = 3;

    jedna.dalsi = &dva;
}
```



Příklad

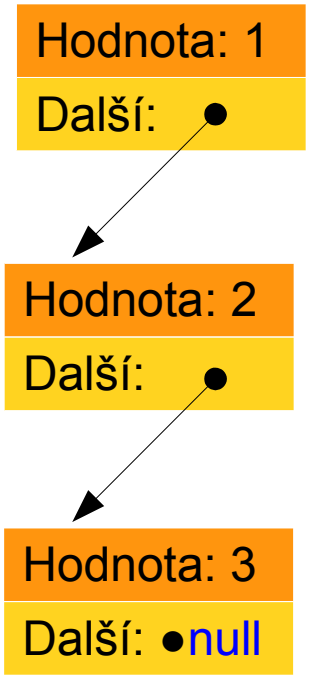
- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

```
public void vytvor123(){
    Prvek jedna, dva, tri;
    jedna=new Prvek(); jedna.hodnota=1;
    dva=new Prvek(); dva.hodnota=2;
    tri=new Prvek(); tri.hodnota=3;

    jedna.dalsi=dva;
    dva.dalsi=tri;
    tri.dalsi=null;
}
```

```
void vytvor123(){
    Prvek jedna, dva, tri;
    jedna.hodnota = 1;
    dva.hodnota = 2;
    tri.hodnota = 3;

    jedna.dalsi = &dva;
    dva.dalsi = &tri;
    tri.dalsi = NULL;
}
```



Příklad

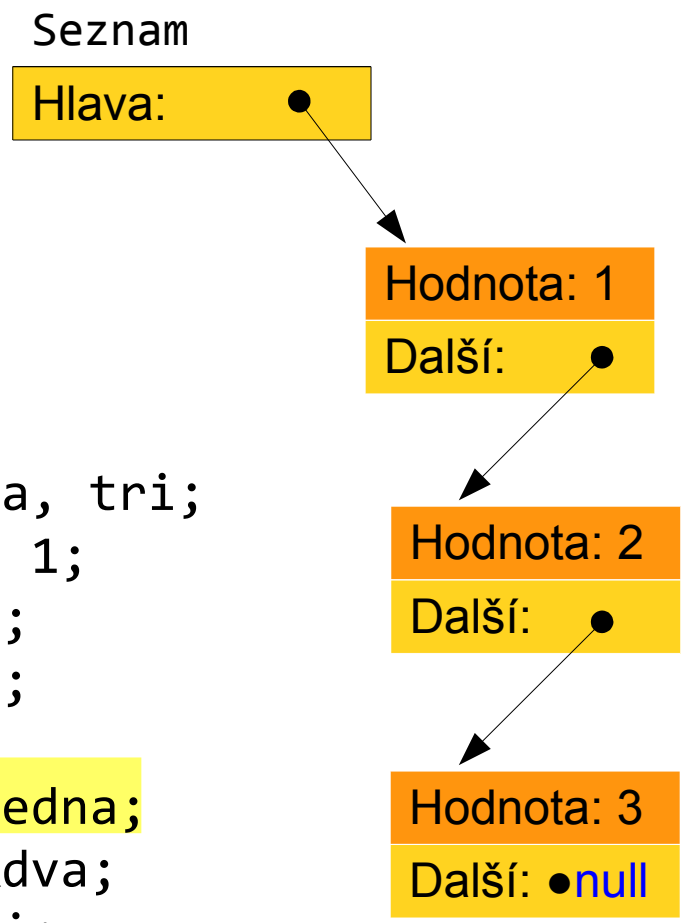
- Vytvořte seznam obsahující prvky s hodnotami 1, 2, 3

```
public void vytvor123(){
    Prvek jedna, dva, tri;
    jedna=new Prvek(); jedna.hodnota=1;
    dva=new Prvek(); dva.hodnota=2;
    tri=new Prvek(); tri.hodnota=3;

    this.hlava=jedna;
    jedna.dalsi=dva;
    dva.dalsi=tri;
    tri.dalsi=null;
}
```

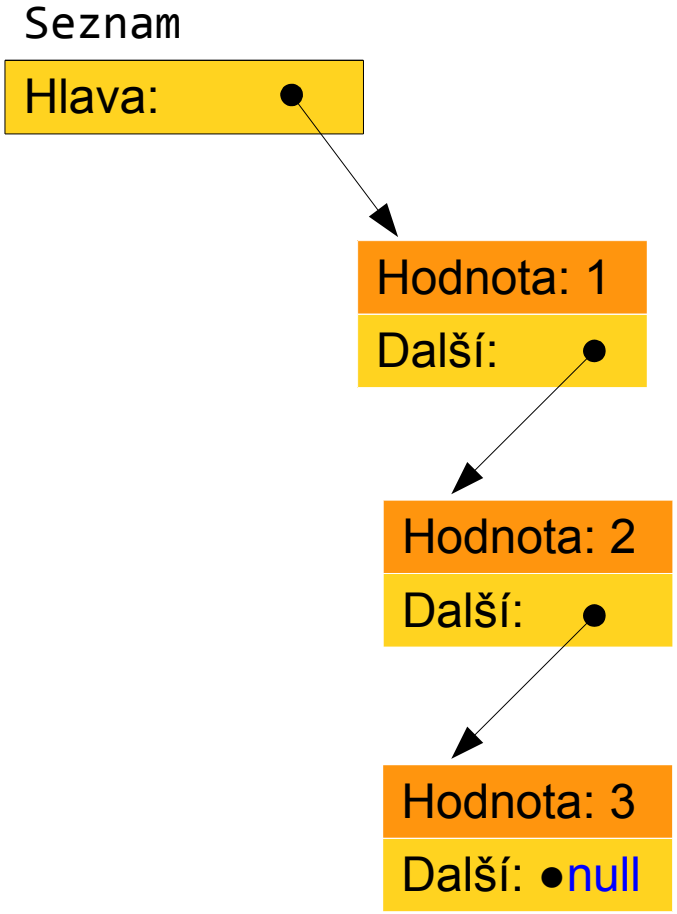
```
void vytvor123(){
    Prvek jedna, dva, tri;
    jedna.hodnota = 1;
    dva.hodnota = 2;
    tri.hodnota = 3;

    this.hlava = &jedna;
    jedna.dalsi = &dva;
    dva.dalsi = &tri;
    tri.dalsi = NULL;
}
```



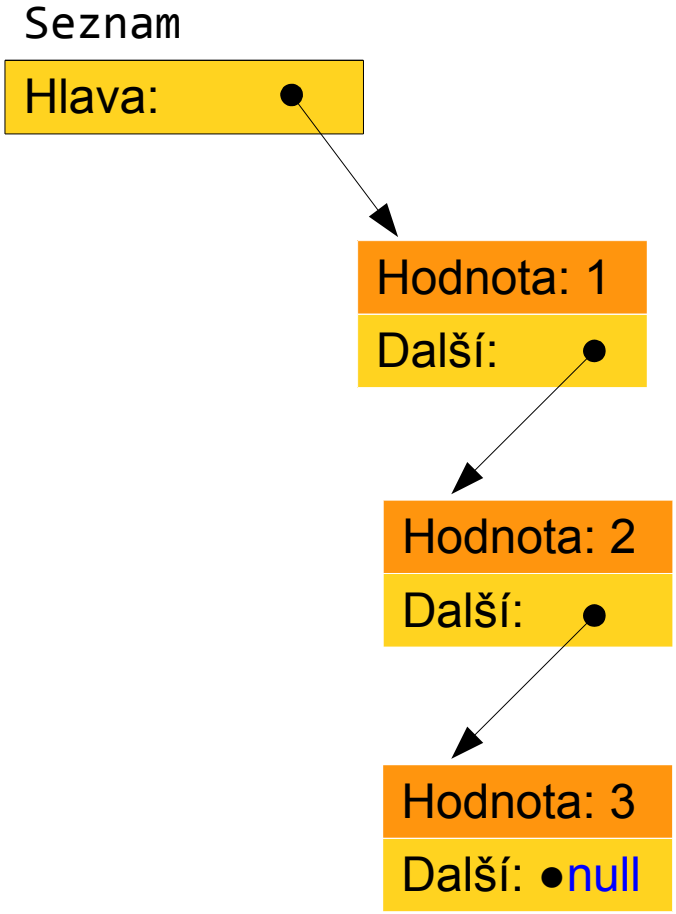
Procházení seznamem

- Začneme v hlavičce a postupně procházíme další prvky seznamu



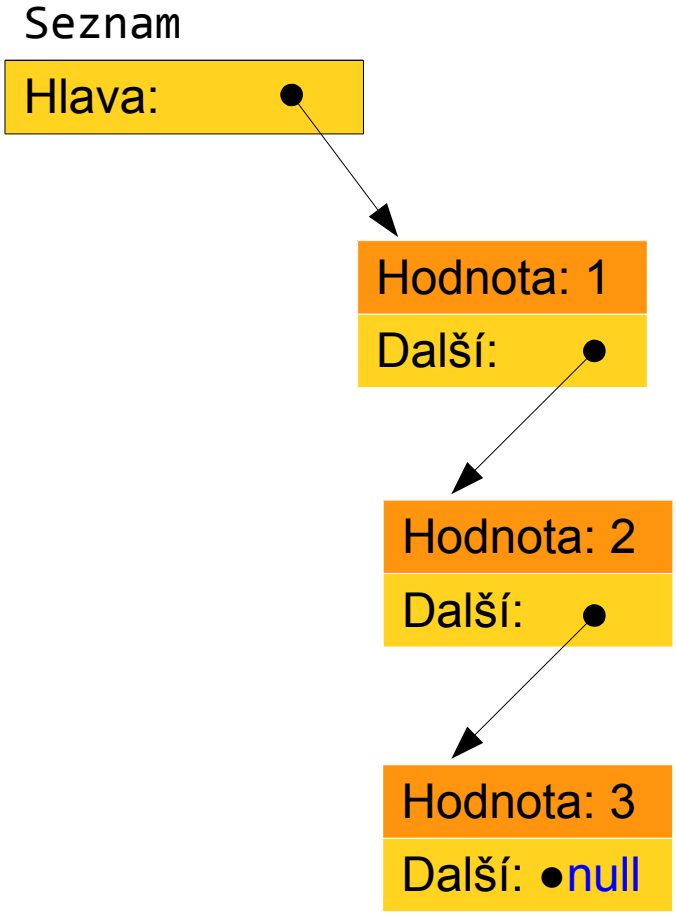
Procházení seznamem

- Začneme v hlavičce a postupně procházíme další prvky seznamu
  - Skončíme, když najdeme ●null
  - Nesmíme ztratit hlavu



Procházení seznamem

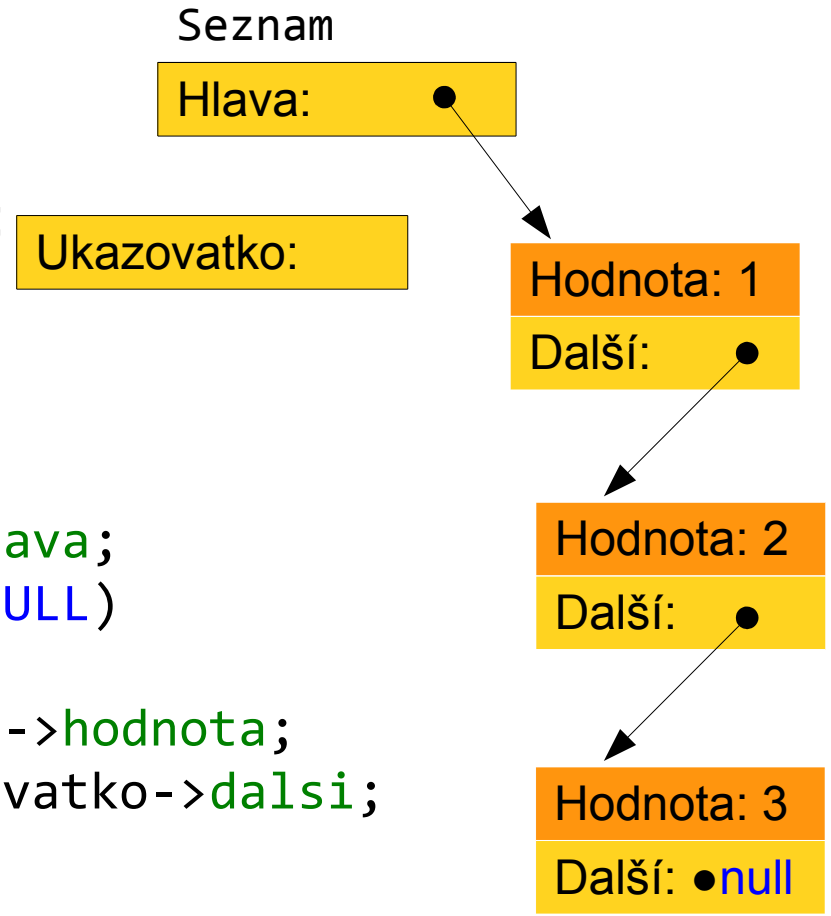
- Začneme v hlavičce a postupně procházíme další prvky seznamu
  - Skončíme, když najdeme ●null
  - Nesmíme ztratit hlavu
    - Použijeme novou proměnnou



Procházení seznamem

- Výpis prvků spojového seznamu

```
public void vypis(){
    Prvek ukazovatko = hlava;
    while (ukazovatko != null)
    {
        System.out.println(ukazovatko.hodnota);
        ukazovatko = ukazovatko.dalsi;
    }
}
```

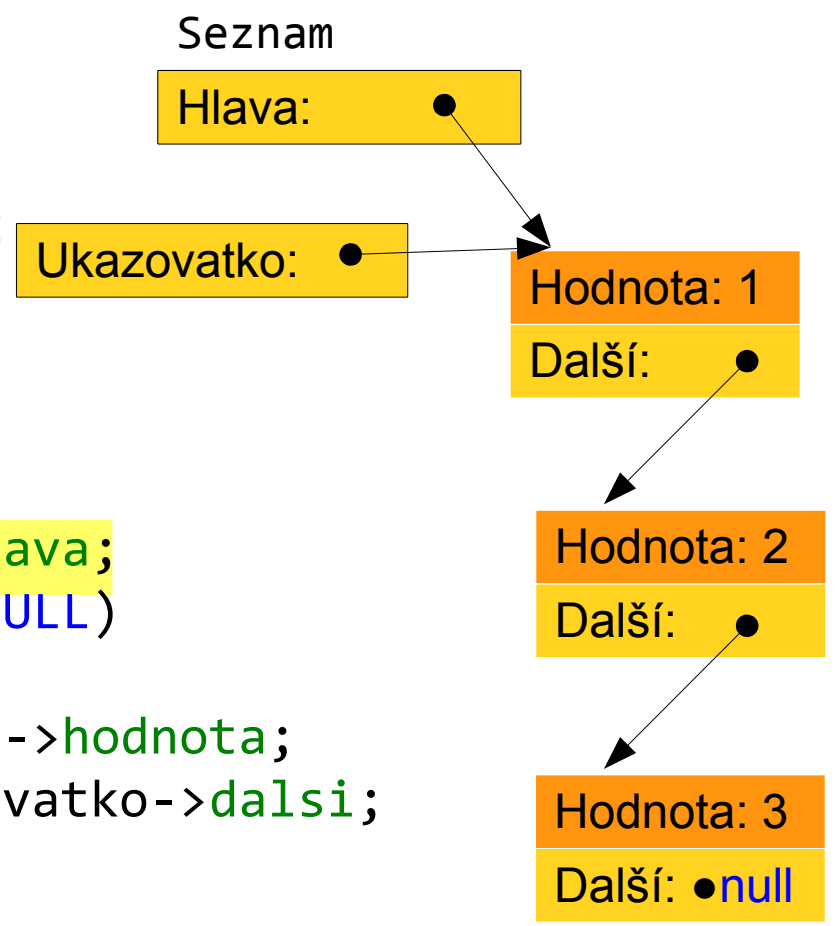



```
void vypis(){
    Prvek* ukazovatko = hlava;
    while (ukazovatko != NULL)
    {
        cout << ukazovatko->hodnota;
        ukazovatko = ukazovatko->dalsi;
    }
}
```

Procházení seznamem

- Výpis prvků spojového seznamu

```
public void vypis(){
    Prvek ukazovatko = hlava;
    while (ukazovatko != null)
    {
        System.out.println(ukazovatko.hodnota);
        ukazovatko = ukazovatko.dalsi;
    }
}
```

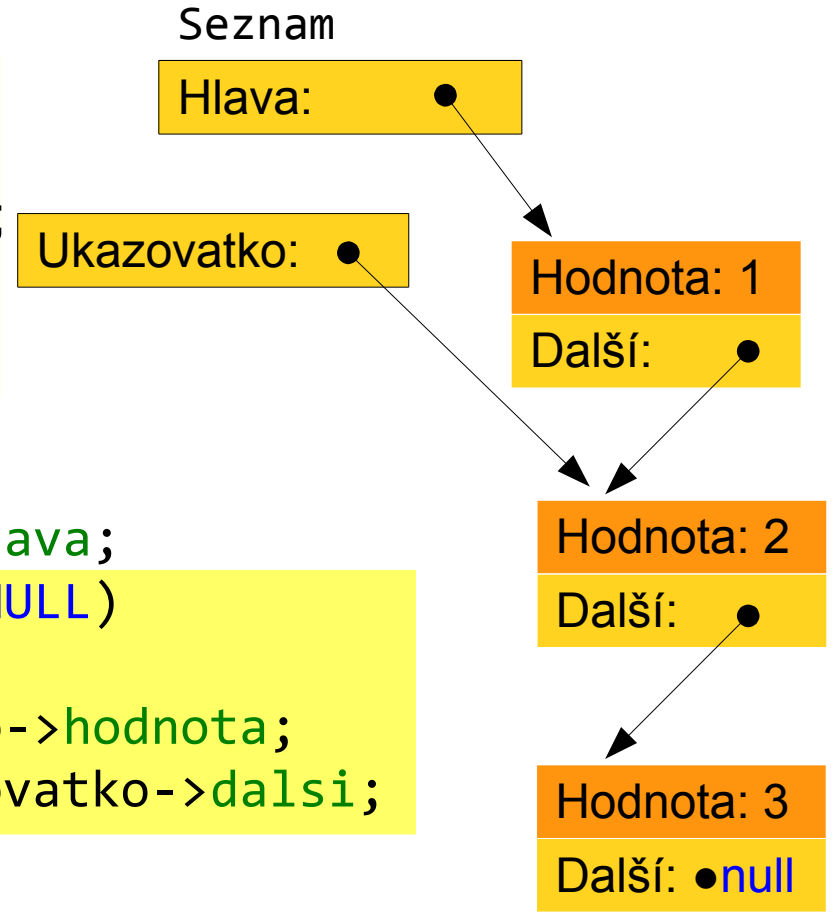



```
void vypis(){
    Prvek* ukazovatko = hlava;
    while (ukazovatko != NULL)
    {
        cout << ukazovatko->hodnota;
        ukazovatko = ukazovatko->dalsi;
    }
}
```

Procházení seznamem

- Výpis prvků spojového seznamu

```
public void vypis(){
    Prvek ukazovatko = hlava;
    while (ukazovatko != null)
    {
        System.out.println(ukazovatko.hodnota);
        ukazovatko = ukazovatko.dalsi;
    }
}
```

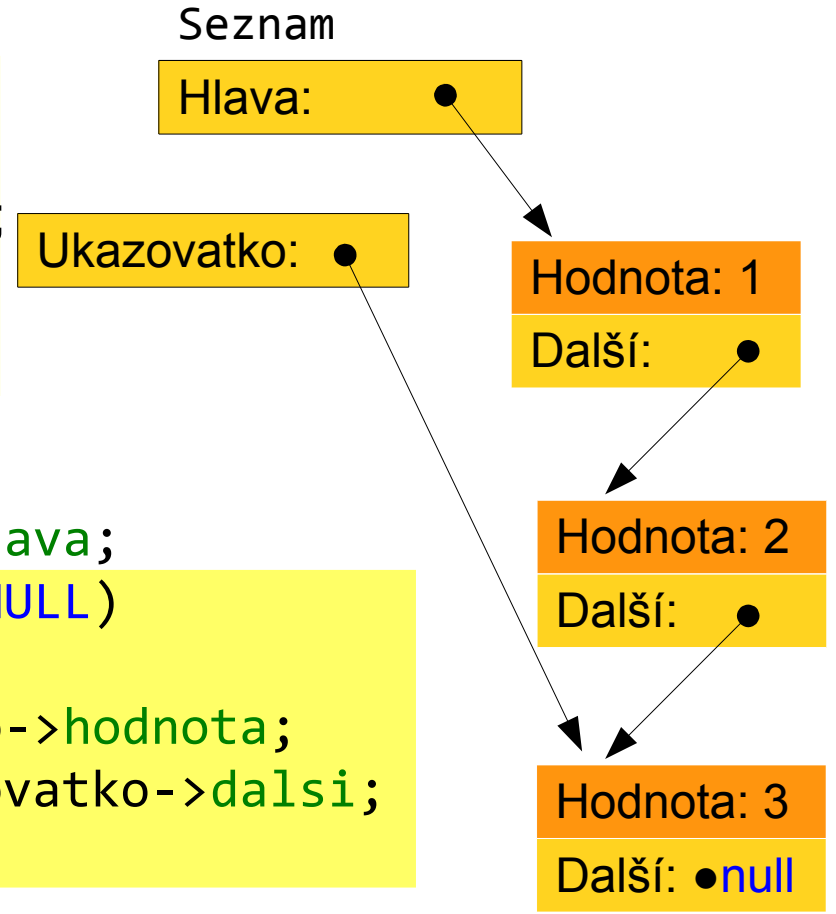



```
void vypis(){
    Prvek* ukazovatko = hlava;
    while (ukazovatko != NULL)
    {
        cout << ukazovatko->hodnota;
        ukazovatko = ukazovatko->dalsi;
    }
}
```

Procházení seznamem

- Výpis prvků spojového seznamu

```
public void vypis(){
    Prvek ukazovatko = hlava;
    while (ukazovatko != null)
    {
        System.out.println(ukazovatko.hodnota);
        ukazovatko = ukazovatko.dalsi;
    }
}
```

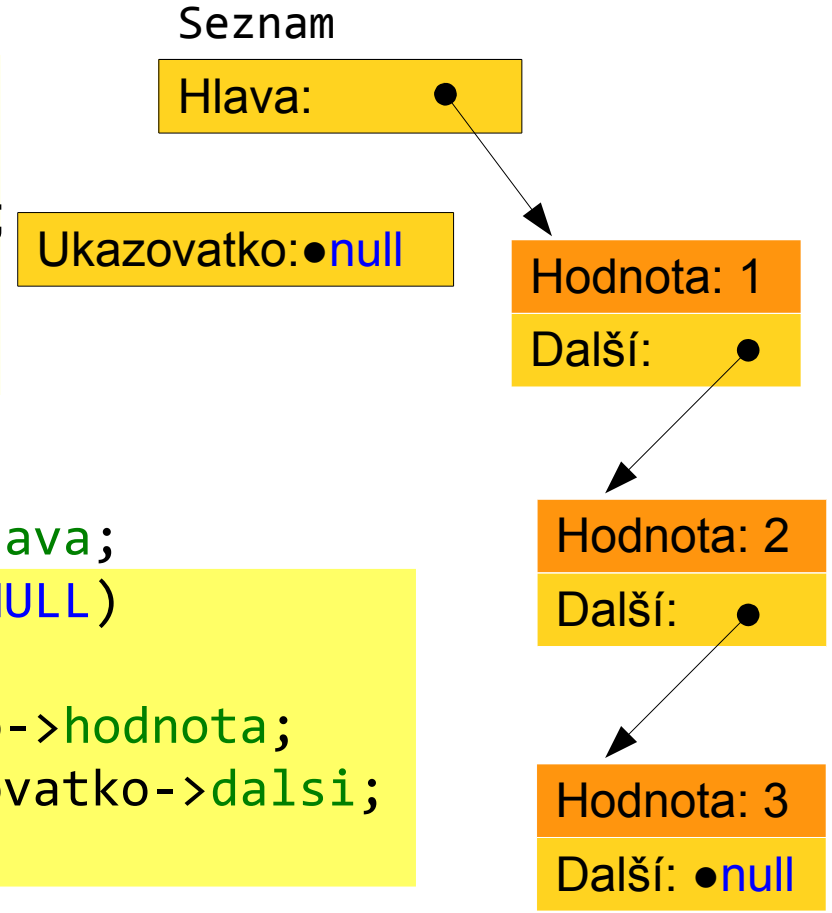



```
void vypis(){
    Prvek* ukazovatko = hlava;
    while (ukazovatko != NULL)
    {
        cout << ukazovatko->hodnota;
        ukazovatko = ukazovatko->dalsi;
    }
}
```

Procházení seznamem

- Výpis prvků spojového seznamu

```
public void vypis(){
    Prvek ukazovatko = hlava;
    while (ukazovatko != null)
    {
        System.out.println(ukazovatko.hodnota);
        ukazovatko = ukazovatko.dalsi;
    }
}
```



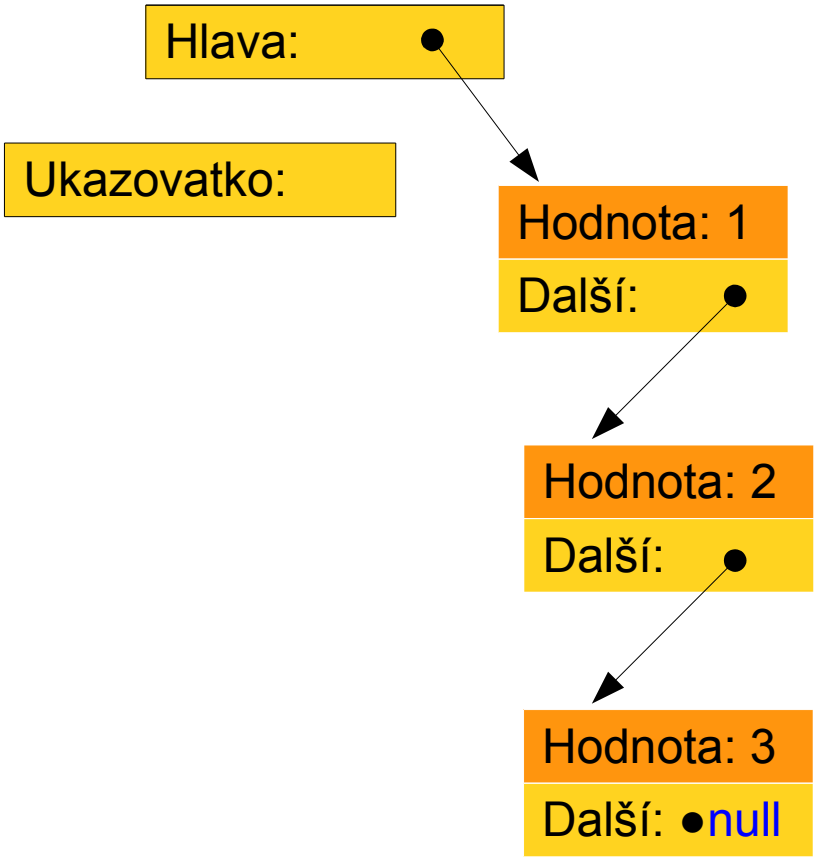

```
void vypis(){
    Prvek* ukazovatko = hlava;
    while (ukazovatko != NULL)
    {
        cout << ukazovatko->hodnota;
        ukazovatko = ukazovatko->dalsi;
    }
}
```

Úkol

- Určete počet prvků v seznamu

Úkol

- Určete počet prvků v seznamu
  - Začneme v hlavičce a postupně procházíme další prvky seznamu, skončíme, když najdeme null

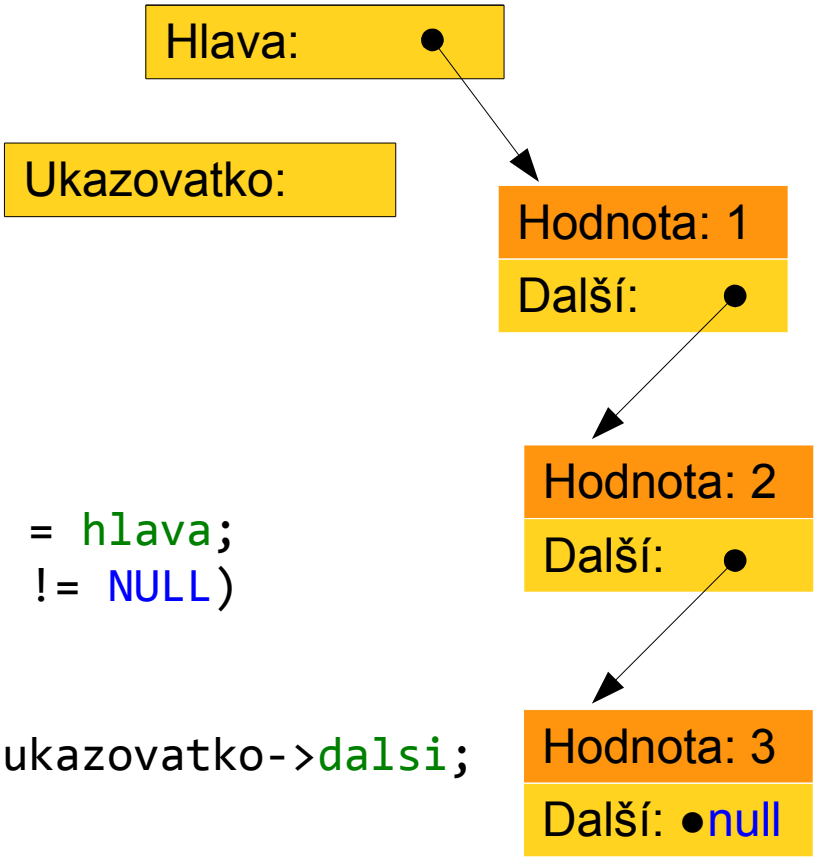


Úkol

- Určete počet prvků v seznamu
  - Začneme v hlavičce a postupně procházíme další prvky seznamu, skončíme, když najdeme null

```
public int PocetPrvku(){
    int pocet=0;
    Prvek ukazovatko = hlava;
    while (ukazovatko != null)
    {
        pocet++;
        ukazovatko = ukazovatko.dalsi;
    }
    return pocet;
}
```

```
int PocetPrvku(){
    int pocet=0;
    Prvek* ukazovatko = hlava;
    while (ukazovatko != NULL)
    {
        pocet++;
        ukazovatko = ukazovatko->dalsi;
    }
    return pocet;
```

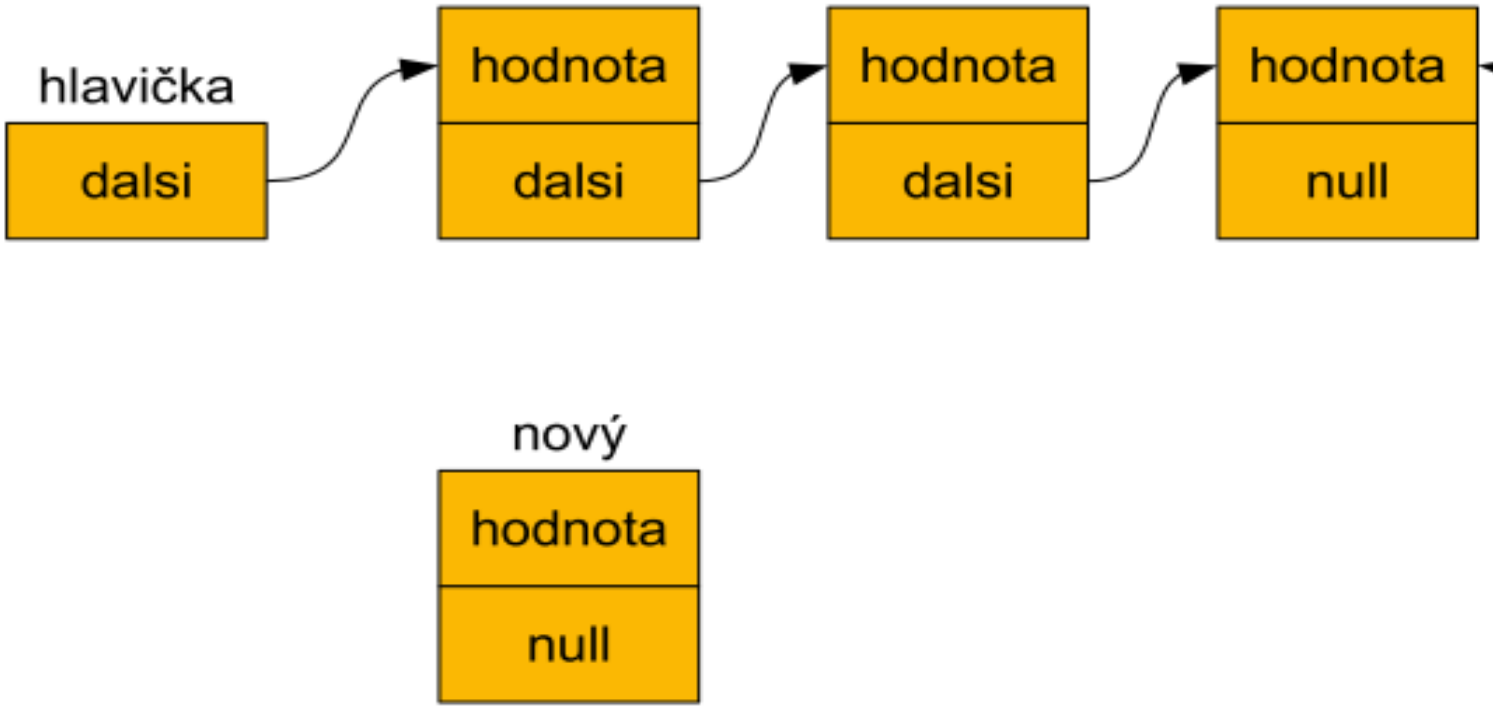


Úkoly

- Zjistěte hodnotu prvního prvku seznamu
- Zjistěte hodnotu druhého prvku seznamu
- Zjistěte hodnotu posledního prvku seznamu
- Určete počet prvků v seznamu
- Nalezněte největší prvek seznamu
- Zjistěte, kolikátý prvek v seznamu má největší hodnotu
- Určete součet všech prvků seznamu
- Zkontrolujte, jestli je seznam setříděný podle velikosti hodnot

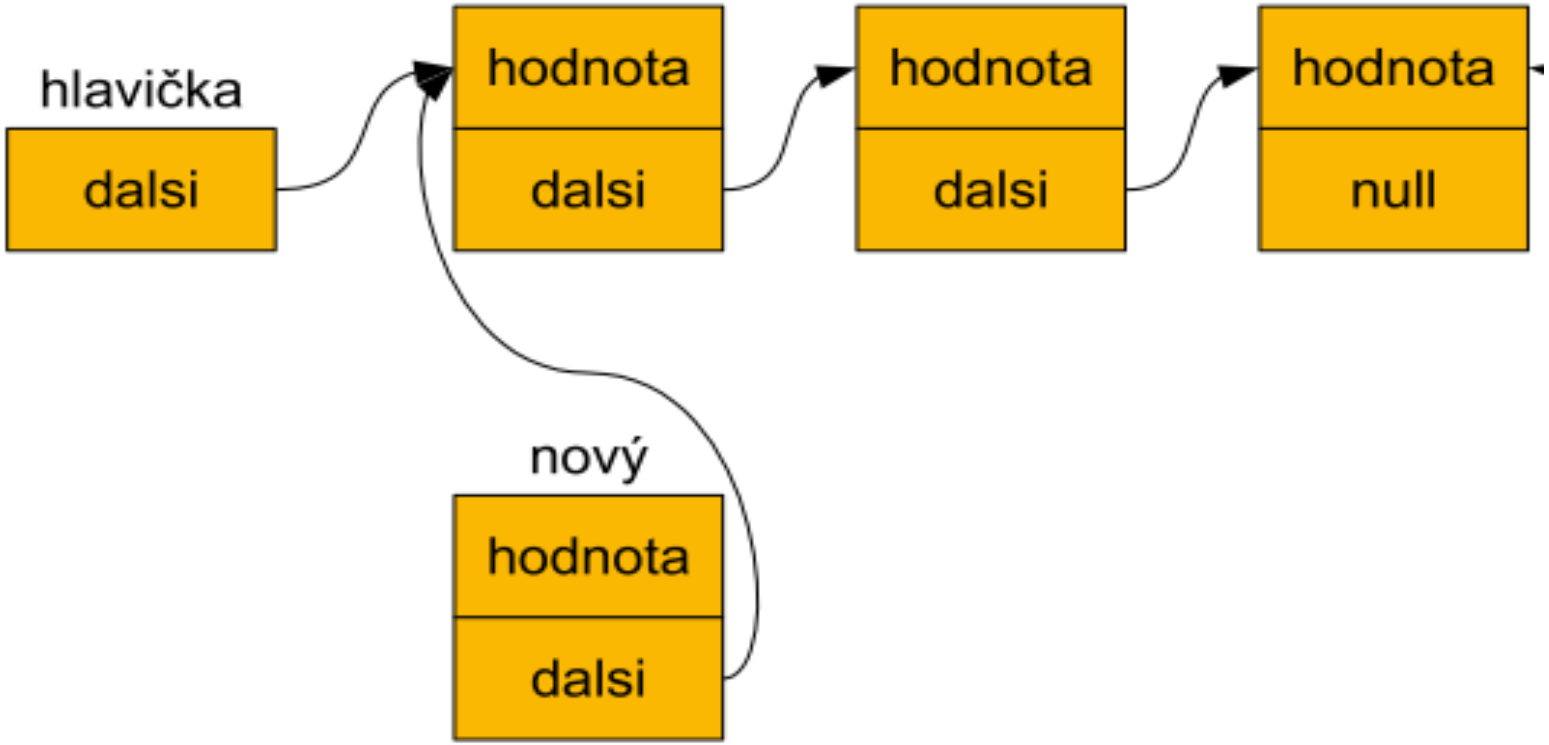
Vložení prvku na začátek seznamu

- Připravíme si prvek, který chceme vkládat



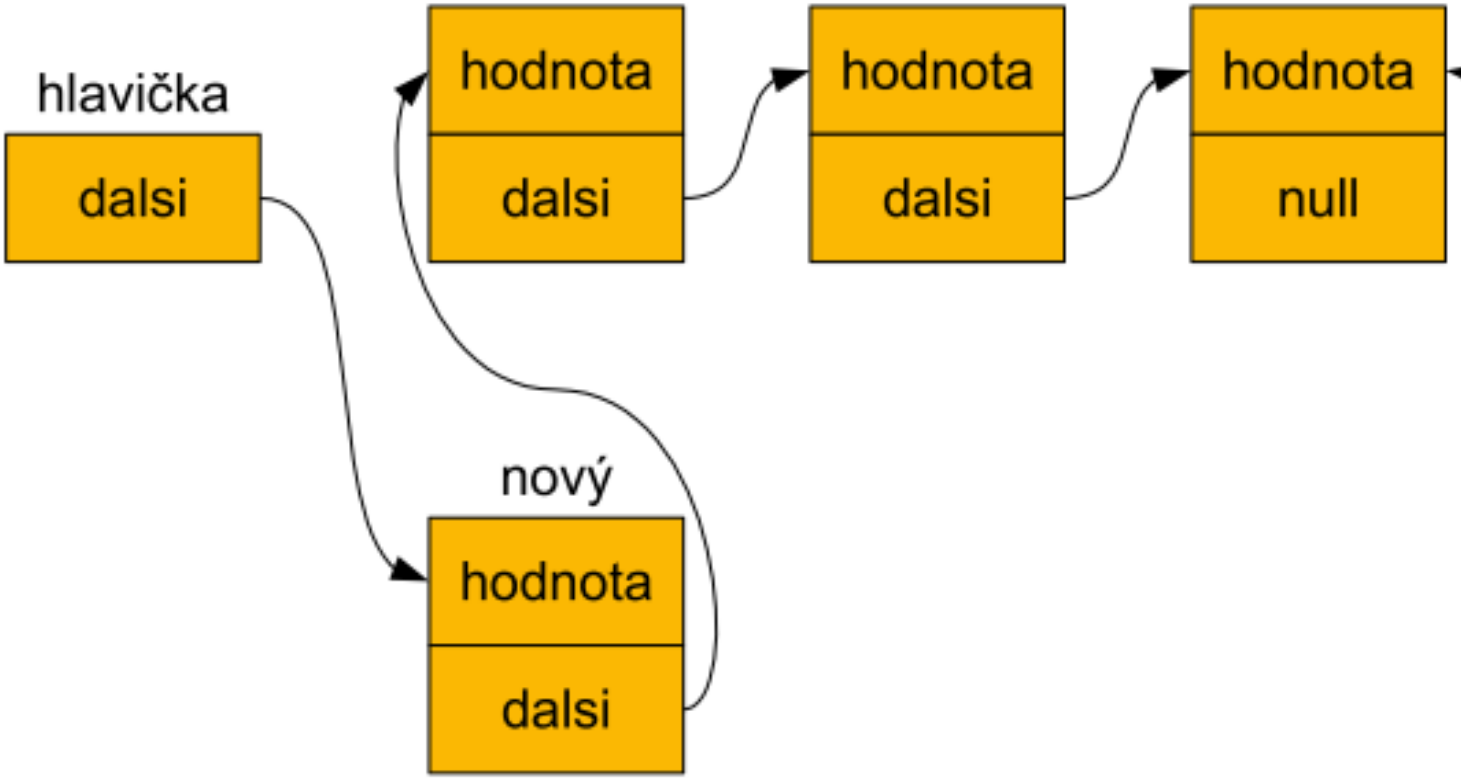
Vložení prvku na začátek seznamu

1. Nastavíme následníka vkládaného na první prvek
   - Nesmíme ztratit zbytek seznamu



Vložení prvku na začátek seznamu

1. Nastavíme následníka vkládaného na první prvek
2. Nastavíme hlavičku na vkládný prvek



Úkol

- Naprogramujte vložení prvku na začátek seznamu
  1. Nastavíme následníka vkládaného na první prvek
  2. Nastavíme hlavičku na vkládný prvek

```
public void vlozNaZacatek(Prvek novy)
{
   novy.dalsi = hlava;
   hlava = novy;
}
```

```
void vlozNaZacatek(Prvek novy)
{
    novy.dalsi = hlava;
    hlava = &novy;
}
```

Úkol

- Naprogramujte vložení prvku na začátek seznamu
  1. Nastavíme následníka vkládaného na první prvek
  2. Nastavíme hlavičku na vkládný prvek

```
public void vlozNaZacatek(Prvek novy)
{
   novy.dalsi = hlava;
   hlava = novy;
}
```

```
void vlozNaZacatek(Prvek novy)
{
    novy.dalsi = hlava;
    hlava = &novy;
}
```

- A co když byl seznam prázdný?

Úkol

- Naprogramujte vložení prvku na začátek seznamu
  1. Nastavíme následníka vkládaného na první prvek
  2. Nastavíme hlavičku na vkládný prvek

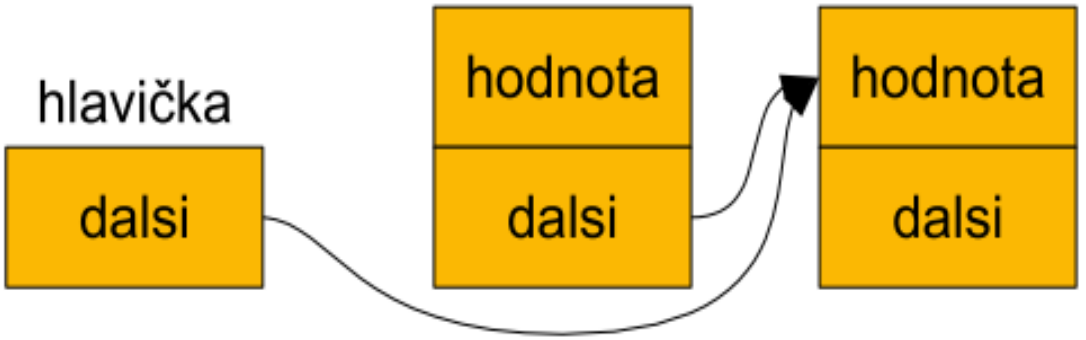
```
public void vlozNaZacatek(Prvek novy)
{
   novy.dalsi = hlava;
   hlava = novy;
}
```

```
void vlozNaZacatek(Prvek novy)
{
    novy.dalsi = hlava;
    hlava = &novy;
}
```

- A co když byl seznam prázdný?
  - Hlavička ukazuje na null, následník nového je null a hlavička ukazuje na nový prvek - OK

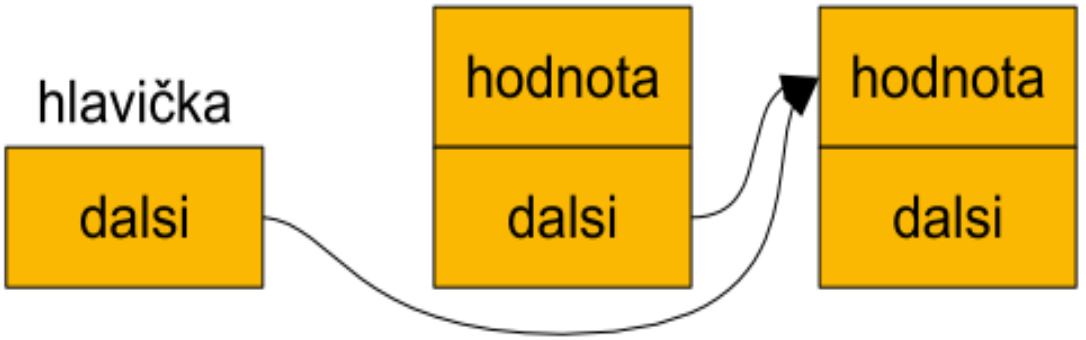
Mazání prvního prvku

- Hlavička má ukazovat na následující prvek



Mazání prvního prvku

- Hlavička má ukazovat na následující prvek




```
public void smazZacatek()
{
   if(hlava!=null)
   {
      hlava = hlava.dalsi;
   }
}
```

```
public void smazZacatek()
{
    if(hlava!=NULL)
    {
        hlava = hlava->dalsi;
    }
}
```

Úkol

- Naprogramujte vložení prvku
  - na začátek
  - na konec
  - za zadaný prvek
  - na i-tou pozici

Úkol

- Naprogramujte smazání prvku
  - Ze začátku
  - z konce
  - zadaného prvku
  - prvku na i-té pozici
  - prvku se zadanou hodnotou

Úkol

- Naprogramujte nalezení
  - prvního prvku
  - posledního prvku
  - prvku se zadanou hodnotou
  - prvku se největší hodnotou
  - prvku na i-té pozici

Úkol

- Otočte směr spojového seznamu

Úkol

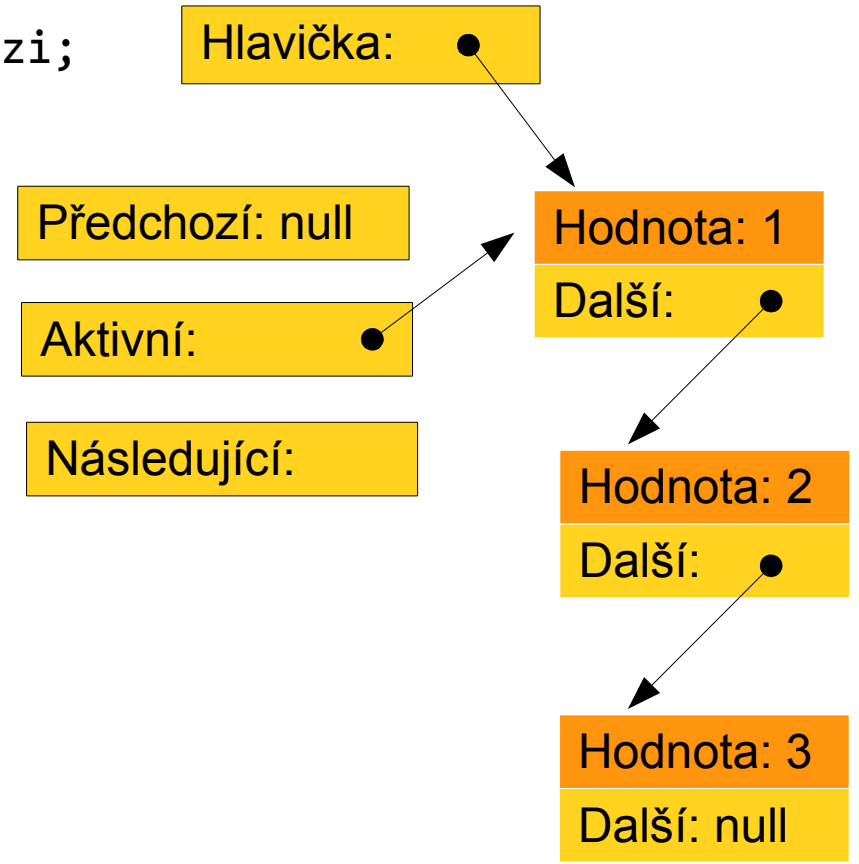
- Otočte směr spojového seznamu

```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```

Úkol

- Otočte směr spojového seznamu

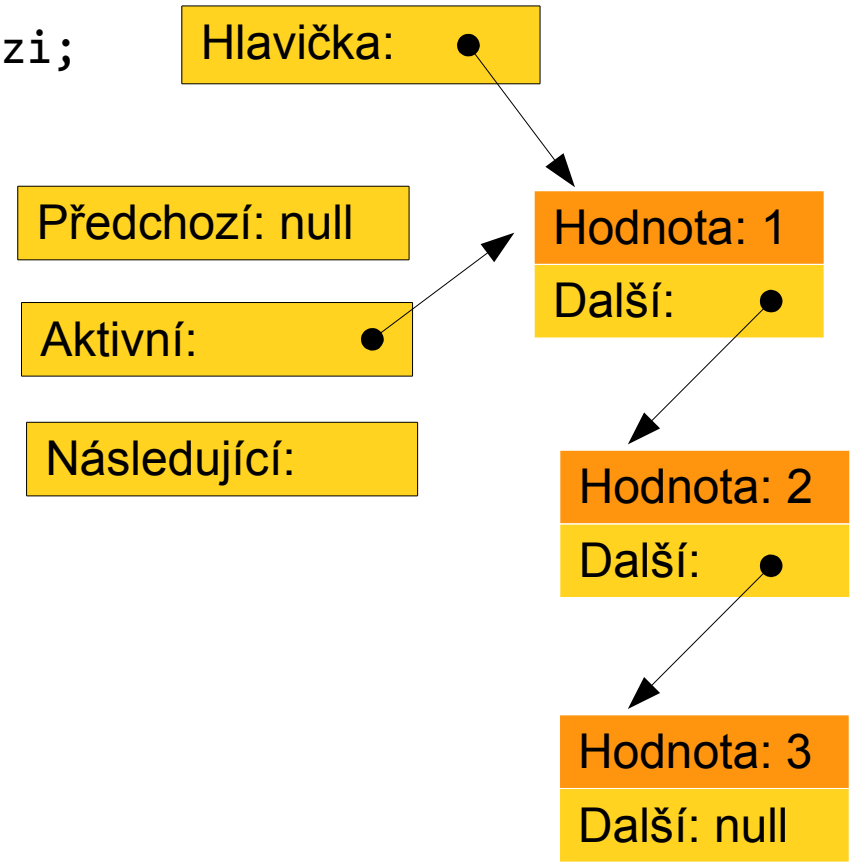
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

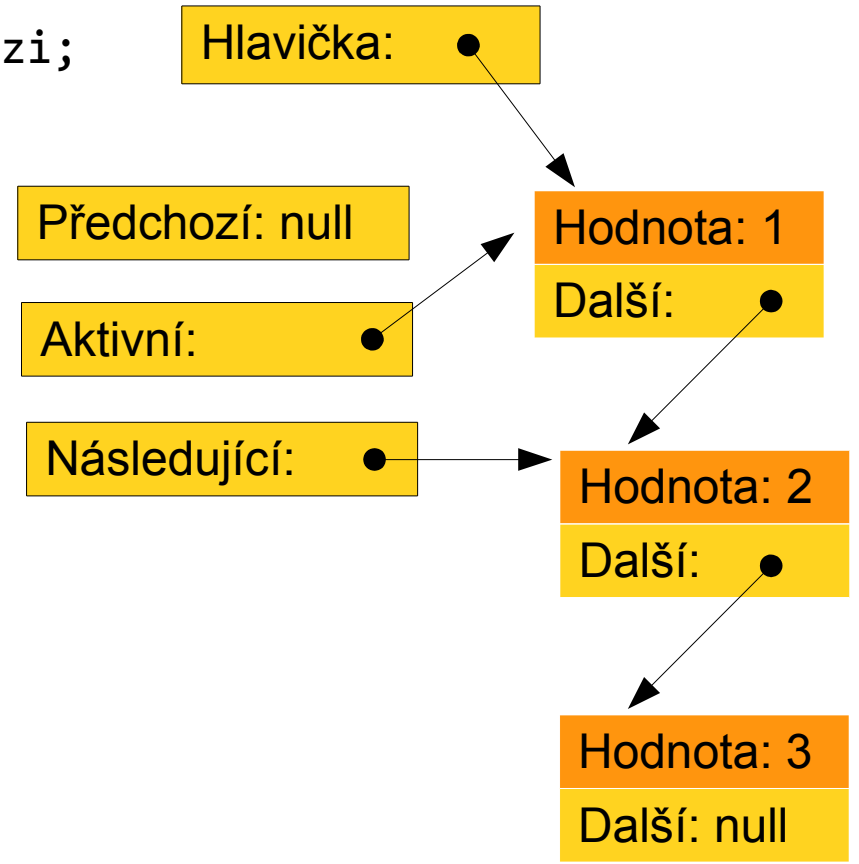
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

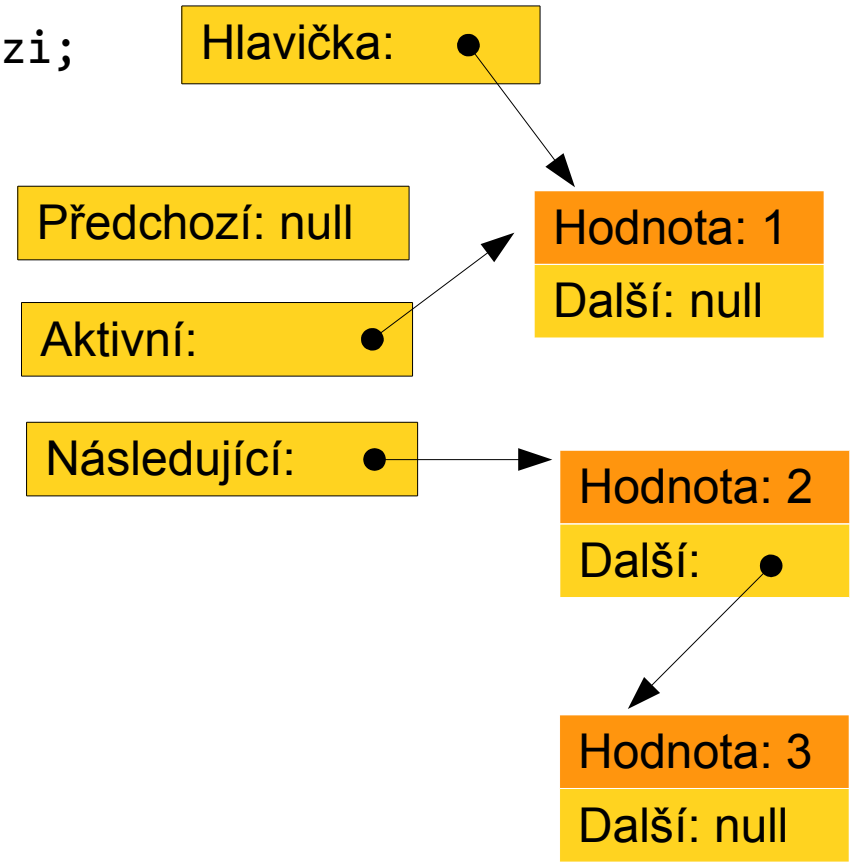
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

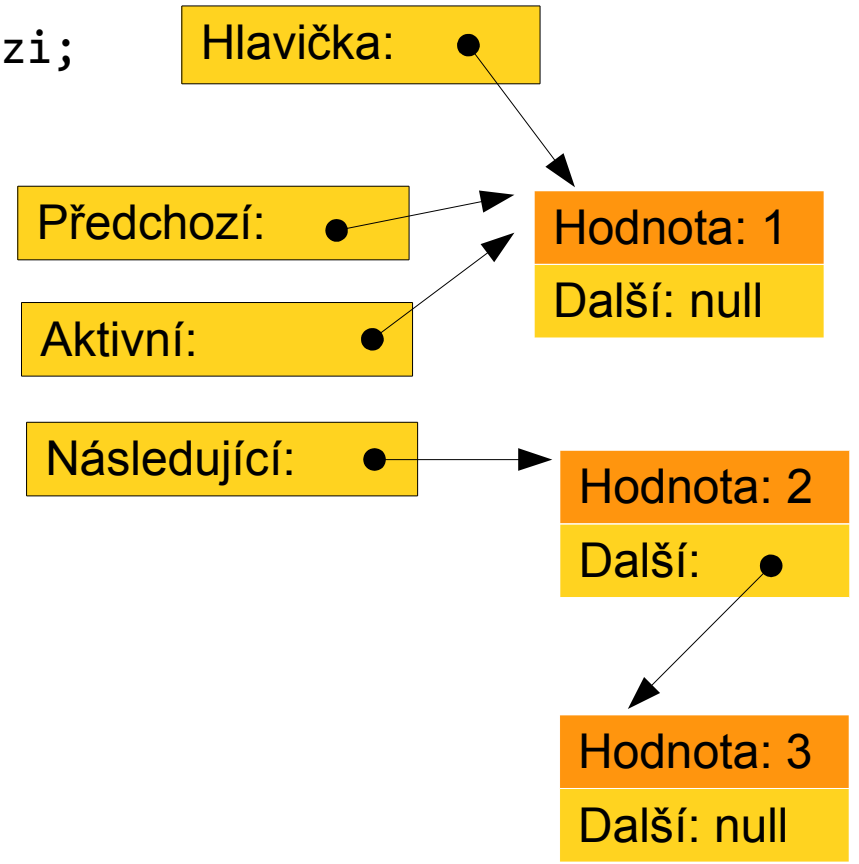
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

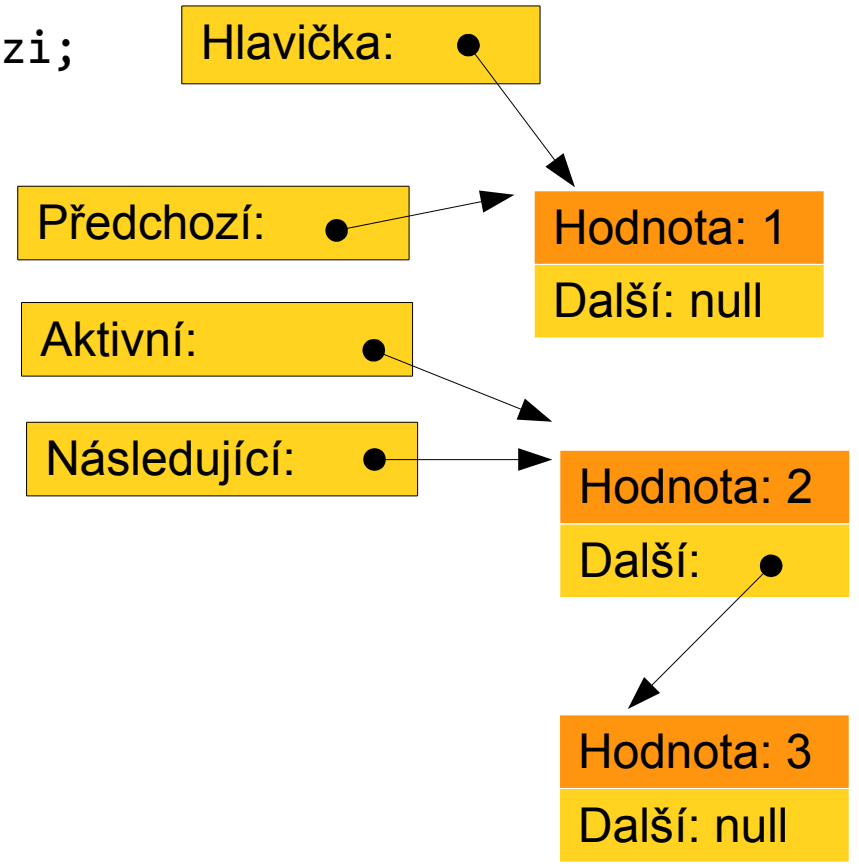
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

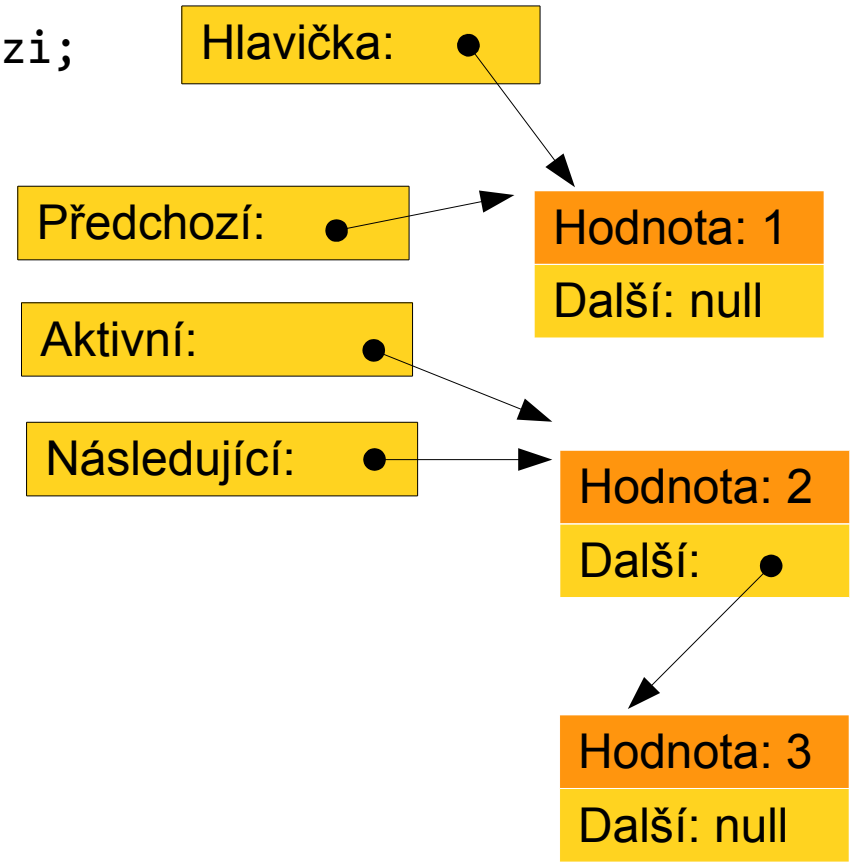
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

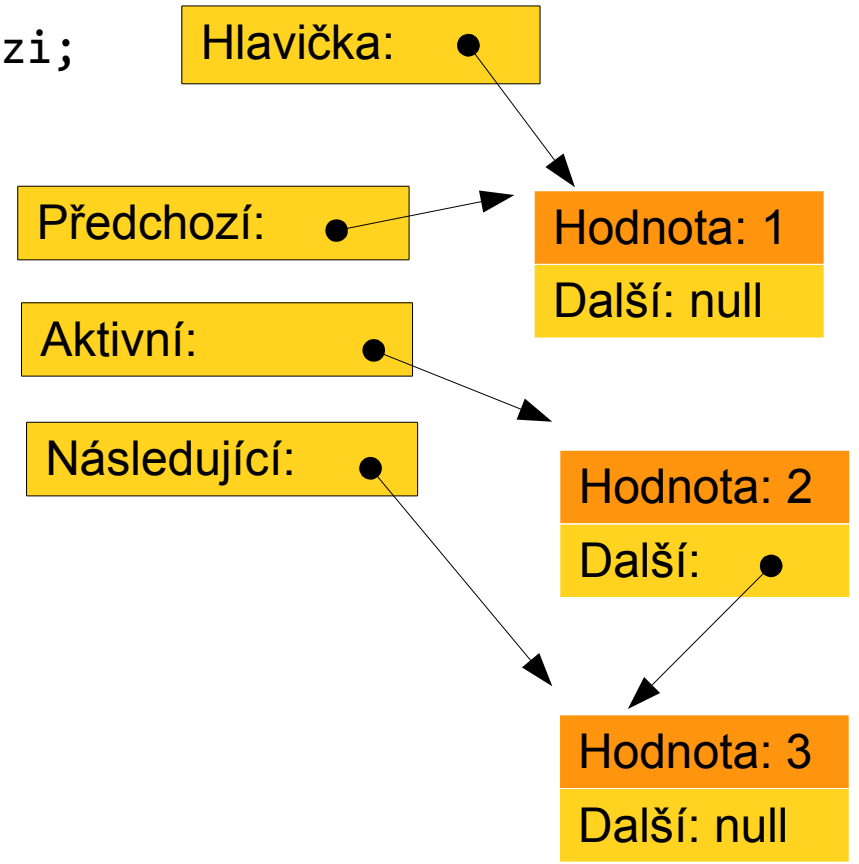
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

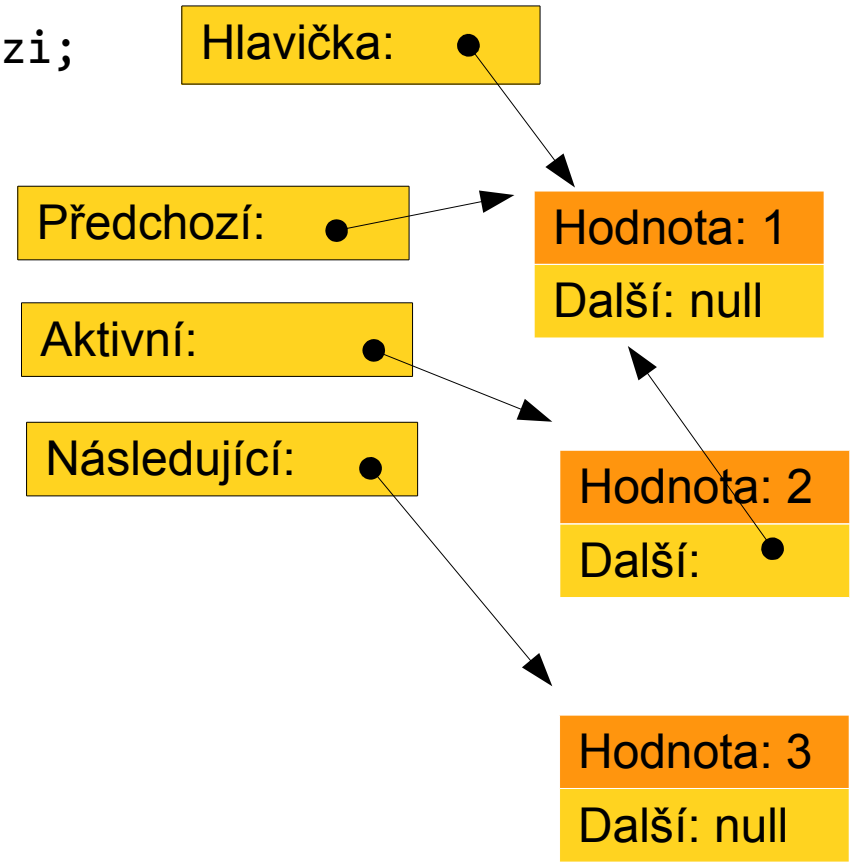
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

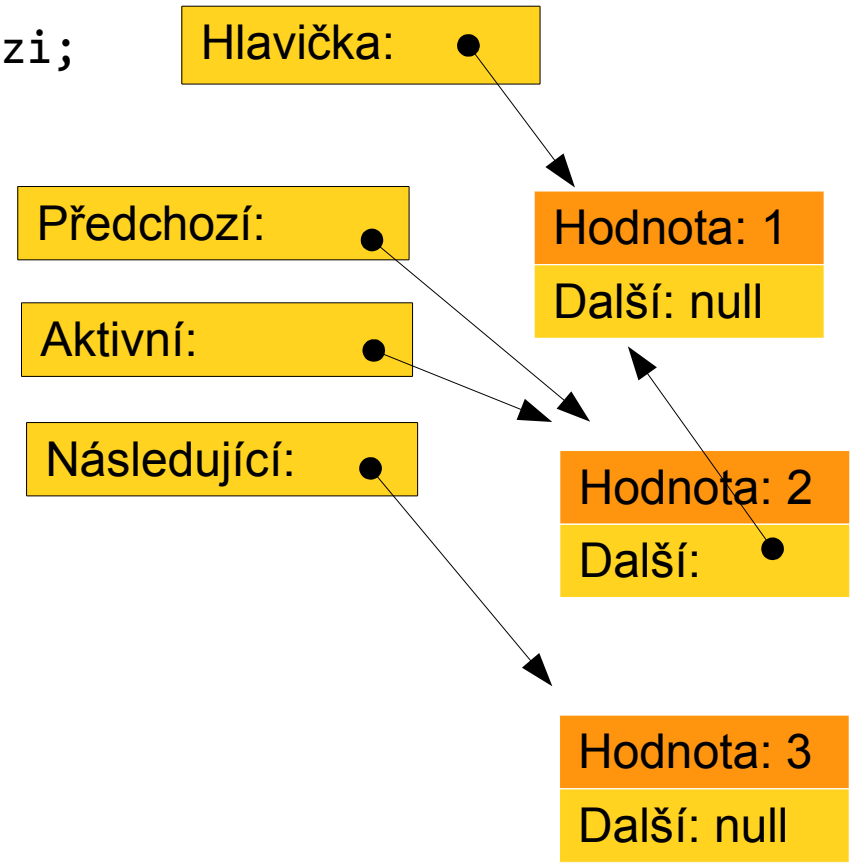
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

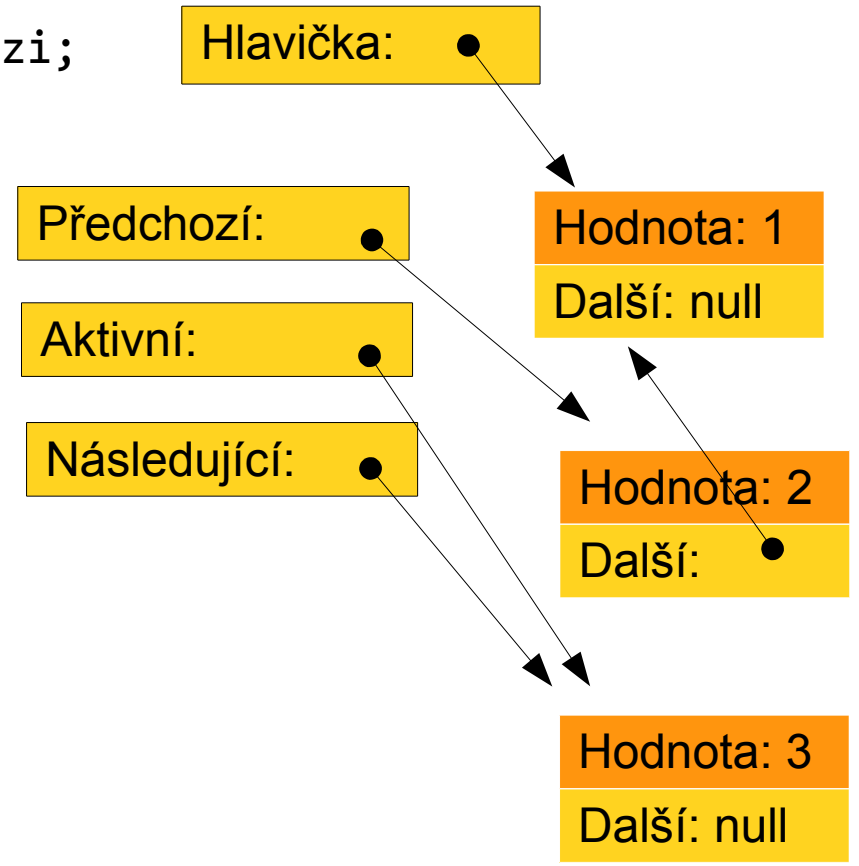
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

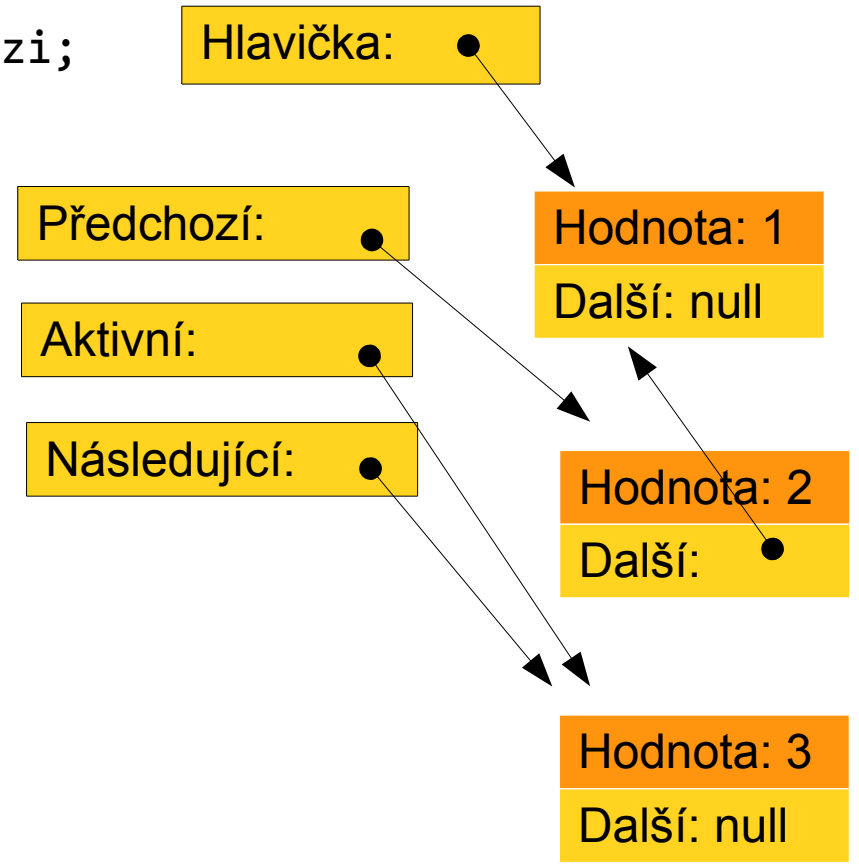
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

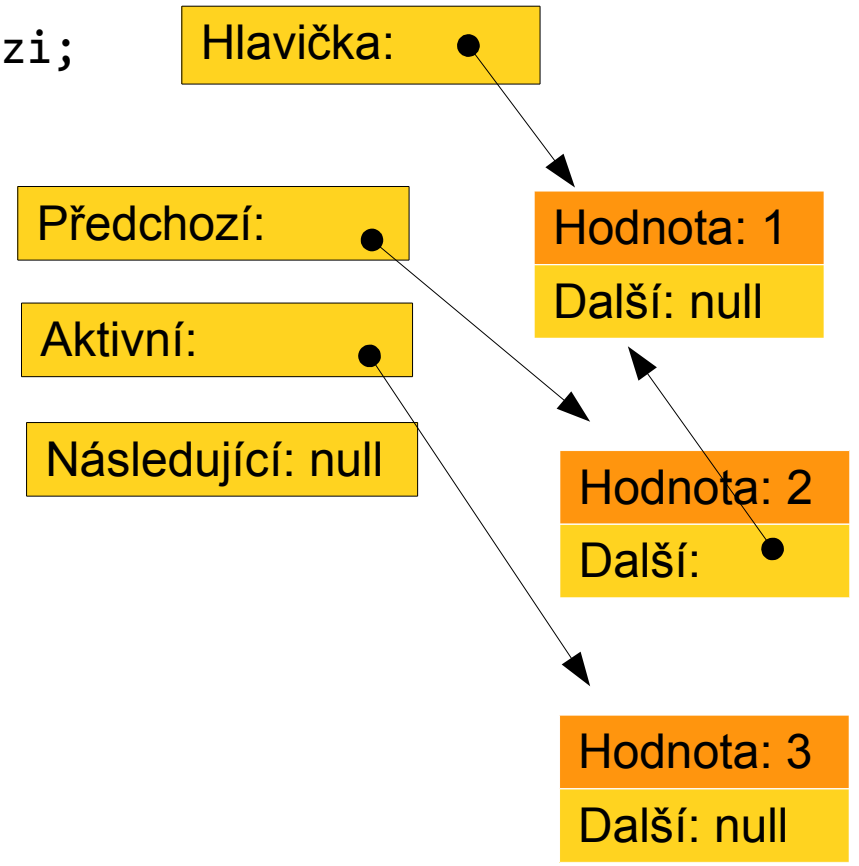
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

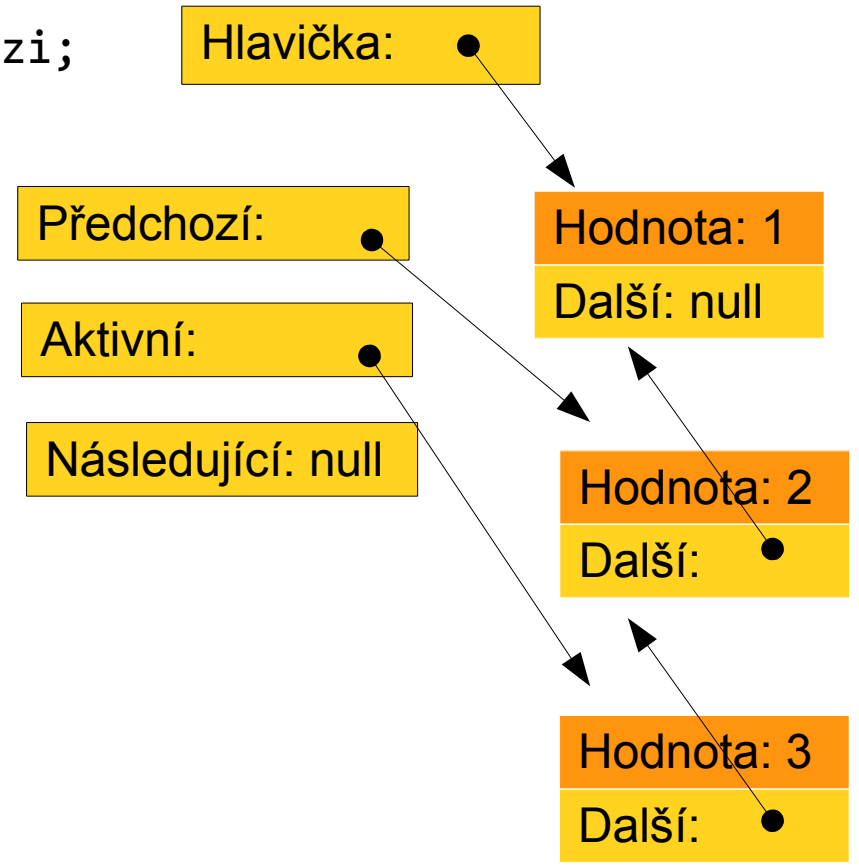
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

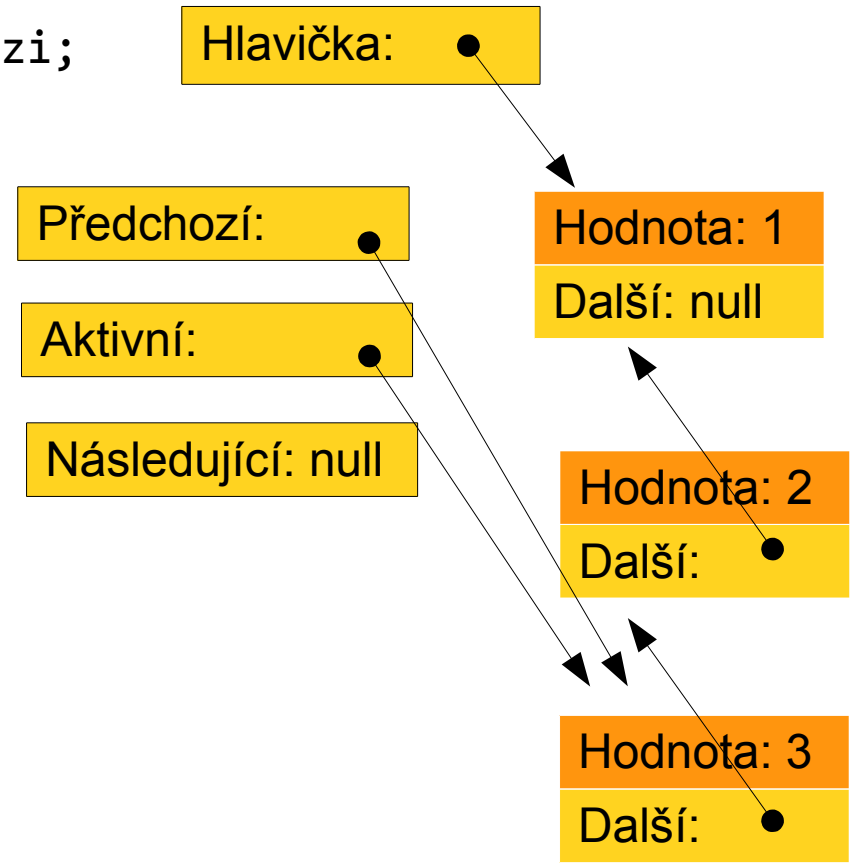
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

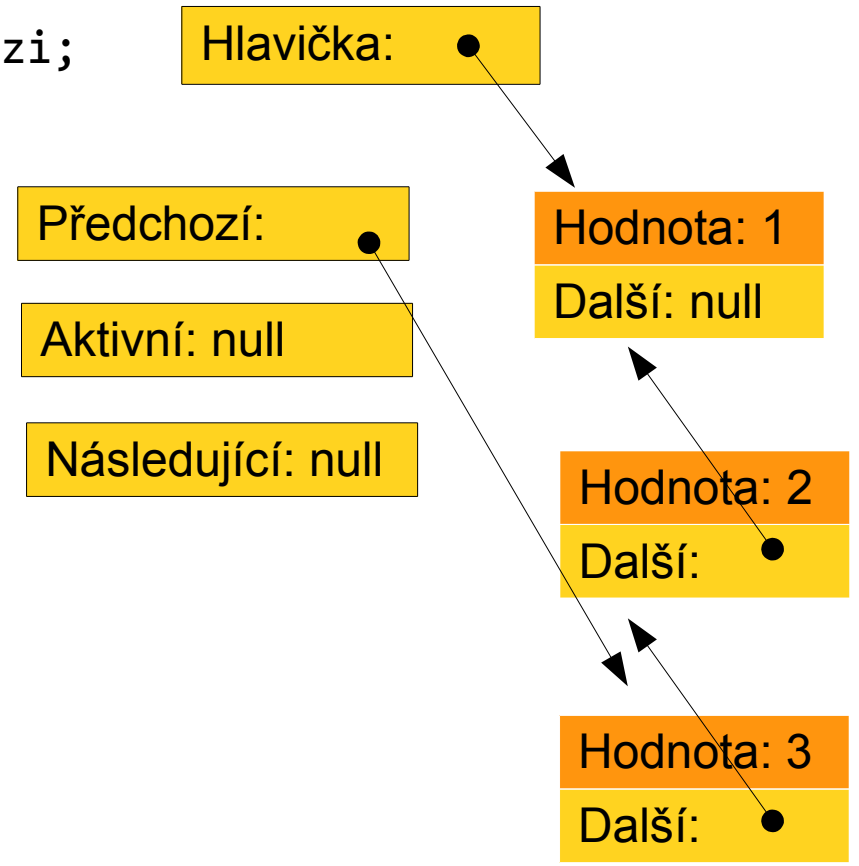
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

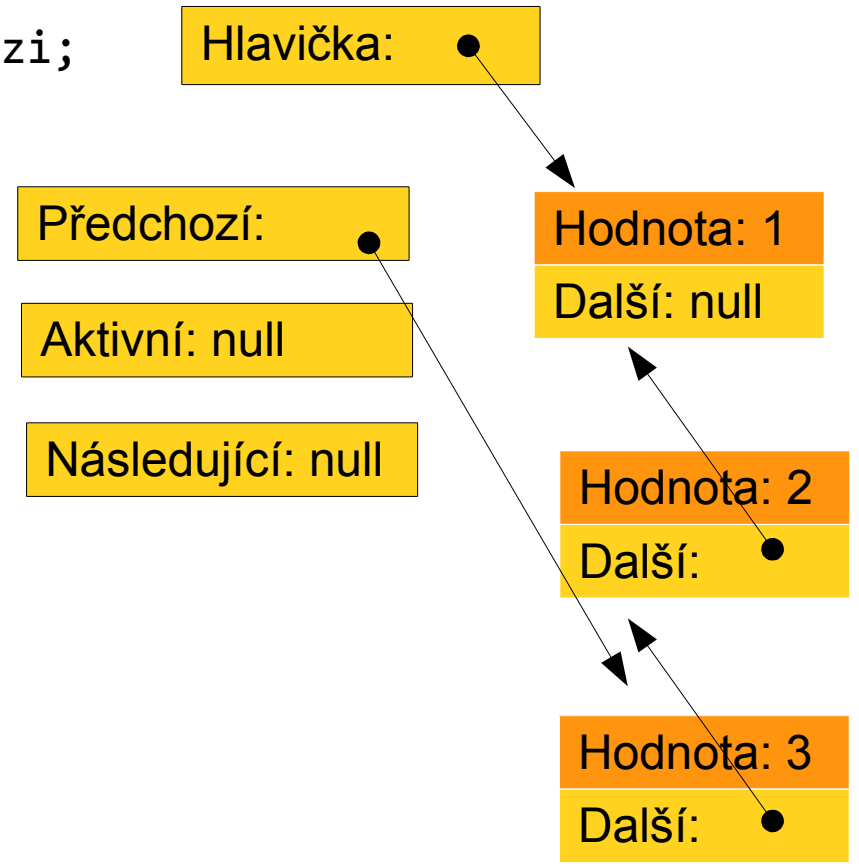
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

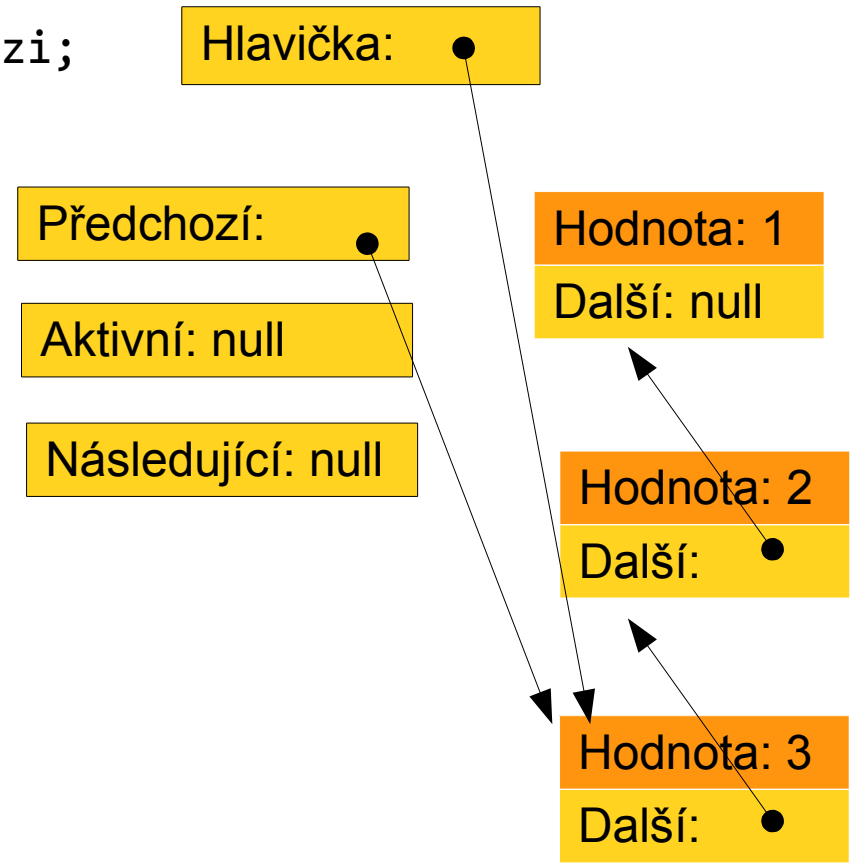
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

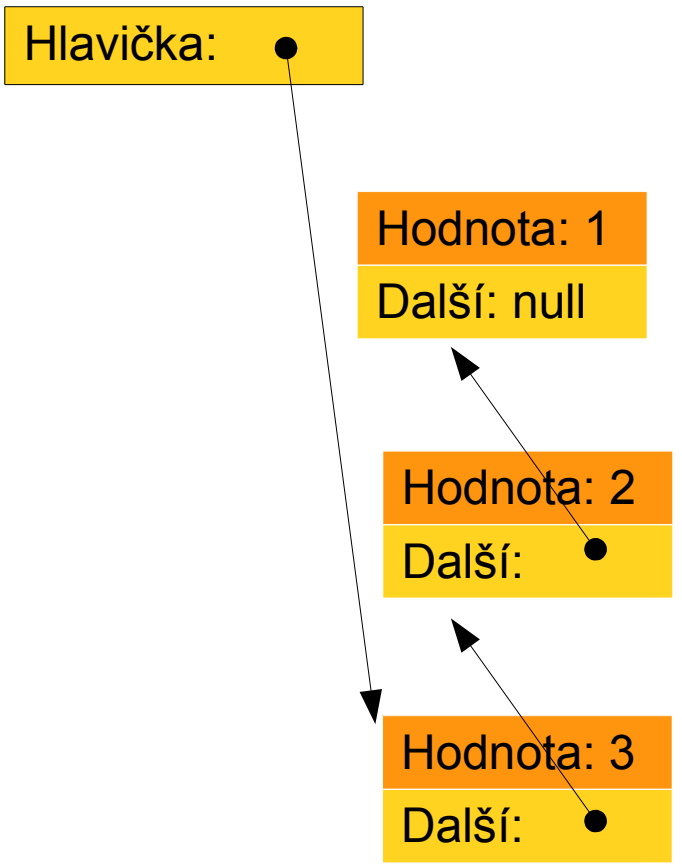
```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Otočte směr spojového seznamu

```
public void otocSmer()
{
    Prvek aktivni, nasledujici, predchozi;
    aktivni = hlavicka;
    predchozi = null;
    while (aktivni != null)
    {
        nasledujici = aktivni.dalsi;
        aktivni.dalsi = predchozi;
        predchozi = aktivni;
        aktivni = nasledujici;
    }
    hlavicka=predchozi;
}
```



Úkol

- Vkládání do spojového seznamu tak, aby prvky byly setříděné
- Sloučení dvou spojových seznamů do jednoho
- Sloučení dvou setříděných spojových seznamů do jednoho setříděného seznamu