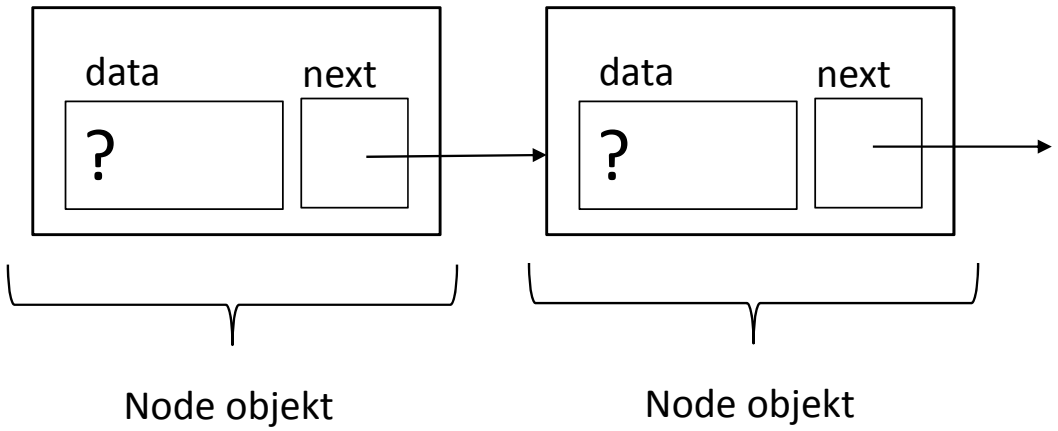
Implementation C++

Enkellänkning med C++

En nod (ett Node-objekt) har adressen till (pekar på) den nod som är dess efterföljaren

```
class Node
{
        T data;
        Node* next;
        …
};
```



Inre klass

- En klass som definieras inom en annan klass
- Tillgänglighet **private** gör att den endast är känd inom den yttre klassen
  - Döljer implementationsdetaljer
  - Den yttre klassen kan deklarera variabler av den inre klasstypen (även pekarvariabler av den inre klasstypen)
  - Den yttre klassen kan instansiera (skapa objekt av) den inre klasstypen

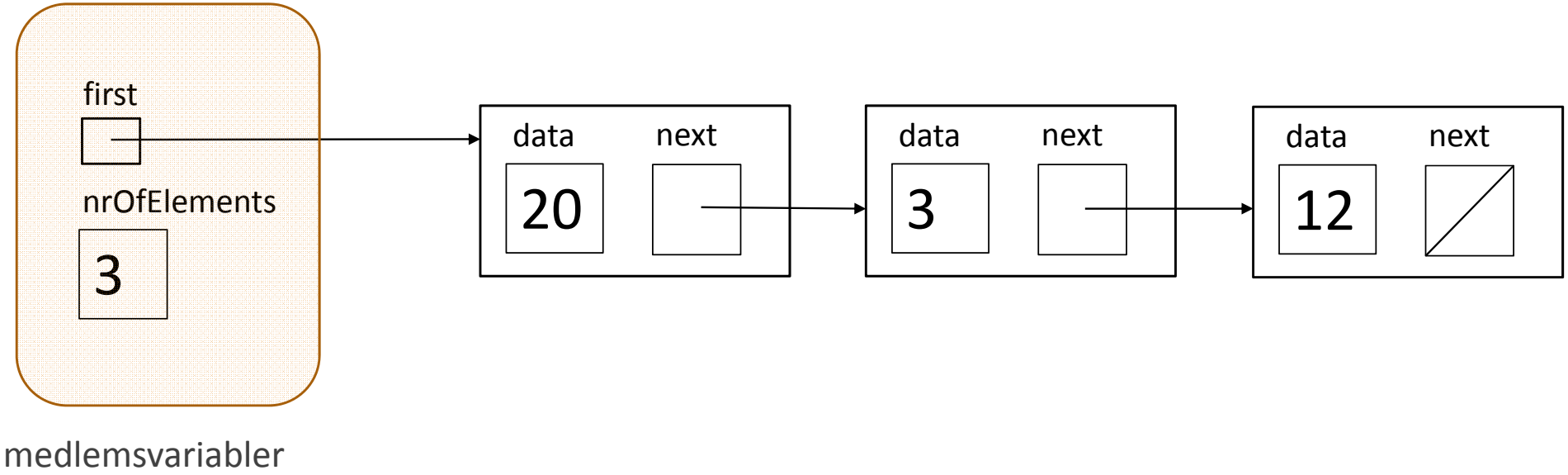
ADT Lista med enkellänkning som intern datastruktur

Filen List.h

```
template <typename T>
class List
{
private:
        class Node
        {
        public:
                T data;
                Node *next;
                Node(T data) { this->data = data; this->next = nullptr;}
                ~Node() {}
        };
        Node* first;
        int nrOfElements;
public:
        //deklaration av medlemsfunktioner, konstruktorer, destruktor, etc
};

//definition av medlemsfunktioner, konstruktorer, destruktor, etc
```

Ett ex. på List-objekt för heltal med 3 noder



medlemsvariabler

Infoga nytt element i listan

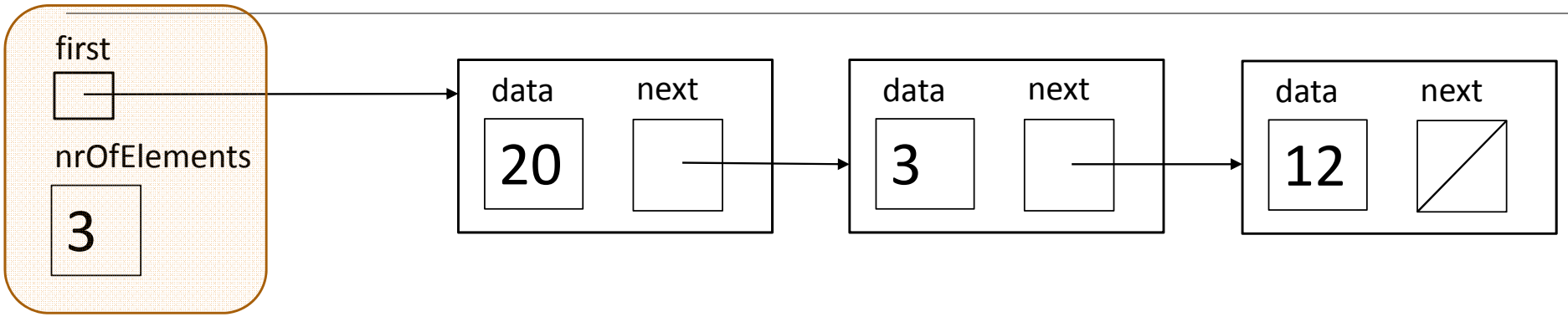
```
template <typename T>
void insertAt(const &T data, int pos) {…}
```

1. Specialfall
   a. pos > nrOfElements – kasta undantag
   b. pos negativt – kasta undantag
   c. pos är 0 - placera först
2. Generella fallet – det nya ska placeras mellan befintliga noder eller som sista nod

pos inte rimlig

```
if (pos < 0 || pos > this->nrOfElements)
        throw ”Exception: pos out of range”;
```

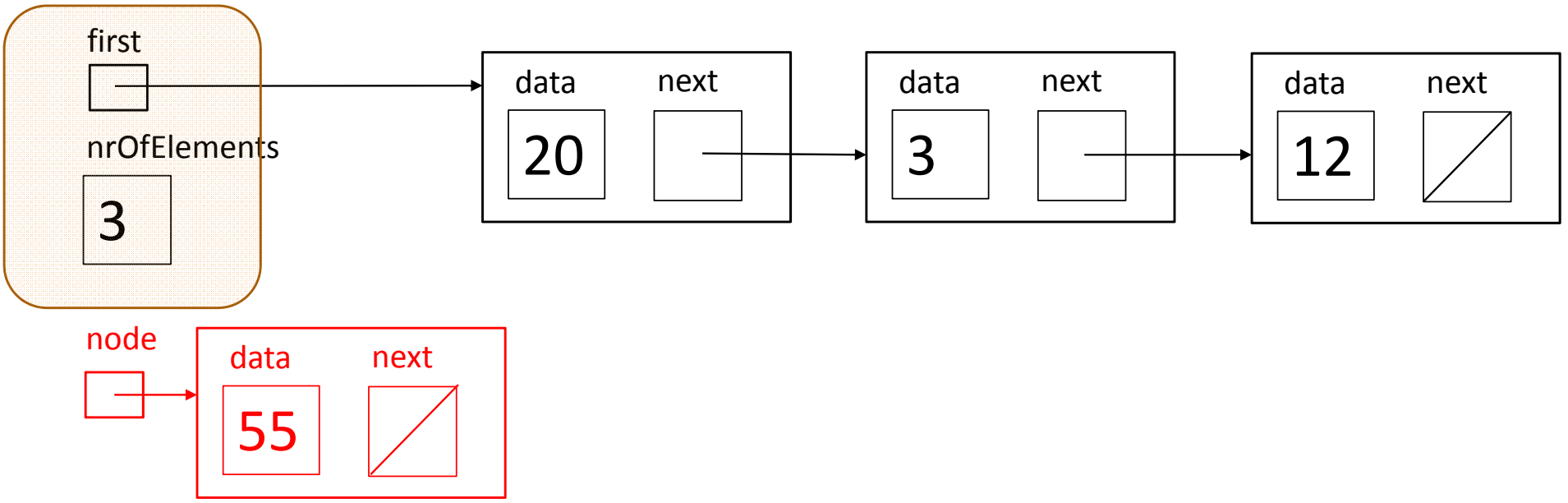
Placera först, if (pos == 0)



1. Skapa en ny nod (ett nytt Node-objekt) för en lokal pekarvariabel
2. Sätt den nya nodens efterföljare (next) till nuvarande första noden
3. Sätt den nya noden som den första, och öka antalet med 1

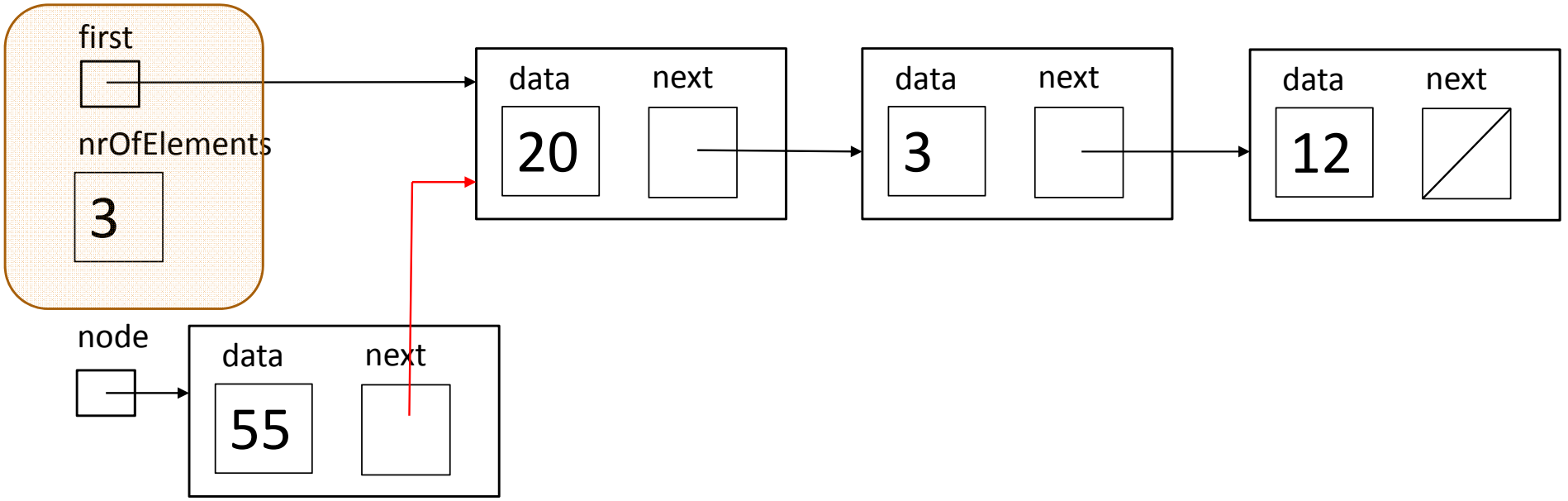
1.Skapa en ny nod (ett nytt Node-objekt) för en lokal pekarvariabel

Node* node = new Node(data);



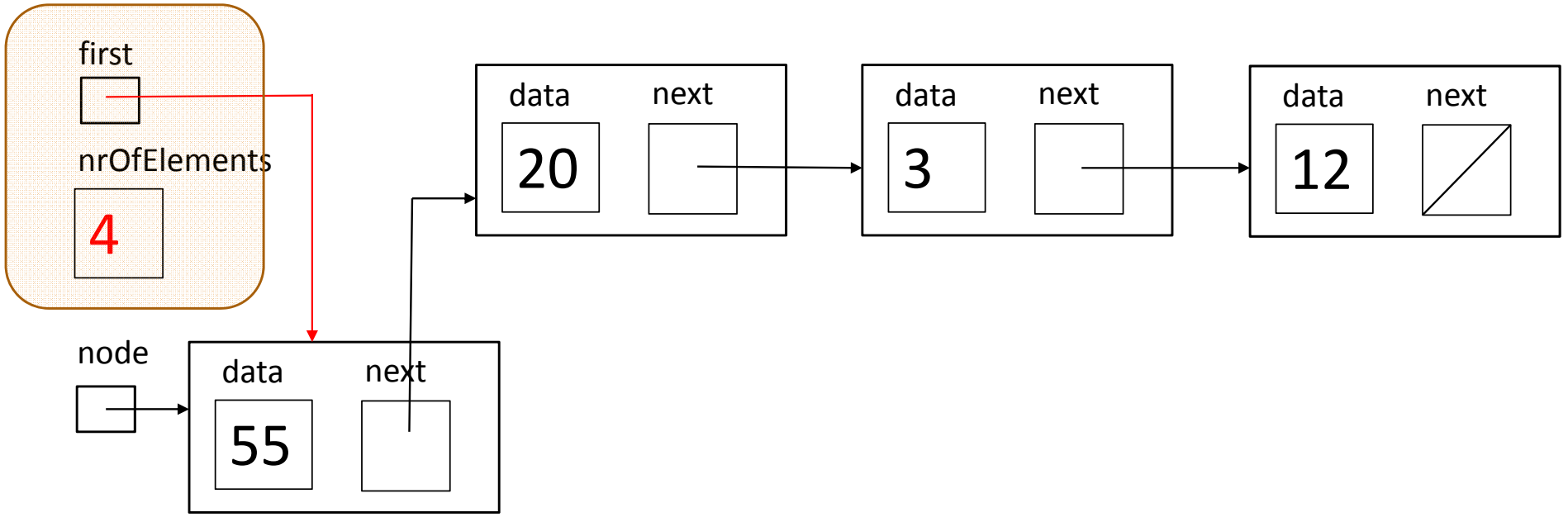
2. Sätt den nya nodens efterföljare (next) till nuvarande första noden

node->next = this->first;

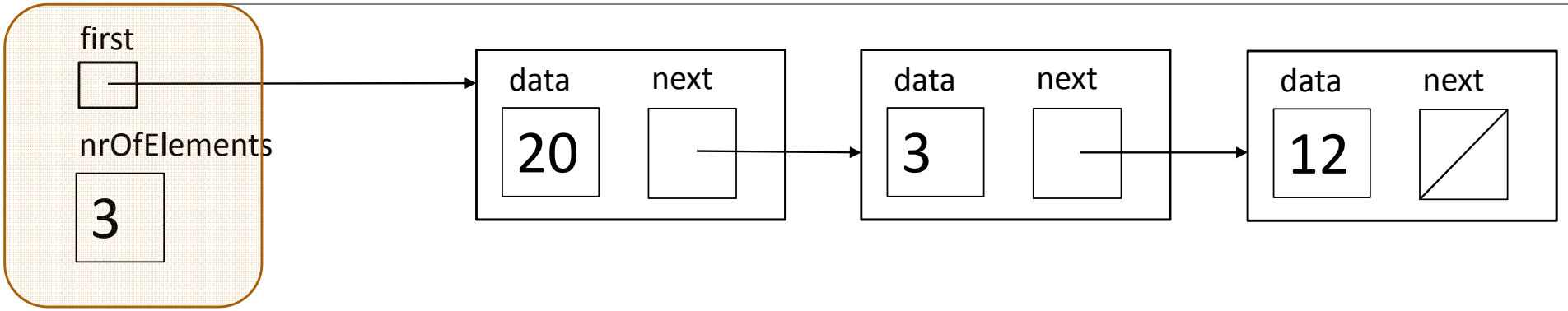


3. Sätt den nya noden som den första, och öka antalet med 1

this->first = node; this->nrOfNodes++;



Placera på den position som pos anger



1. Skapa en ny Node-pekare (som ska användas för att stega framåt) och ge den adressen till första noden
2. ”Flytta” Node-pekaren framåt (byt adressen) så att den slutligen pekar på noden som ska vara ”före” den nya
3. Skapa en ny nod (nytt Node-objekt) och infoga denna efter den nod som Node-pekaren har adressen till och öka antalet

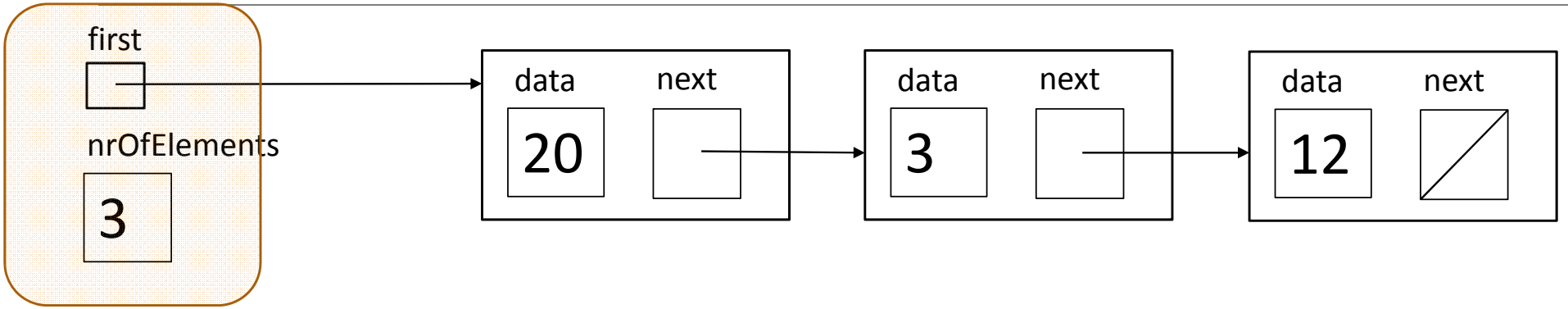
PÅ TAVLAN!

Ta bort ett element från listan

```
template <typename T>
T removeAt(int pos) {…}
```

1. Specialfall
   a. pos > nrOfElements – kasta undantag
   b. pos negativt – kasta undantag
   c. pos är 0 - ta bort första
2. Generella fallet – noden som ska tas bort är andra en nod efter den första

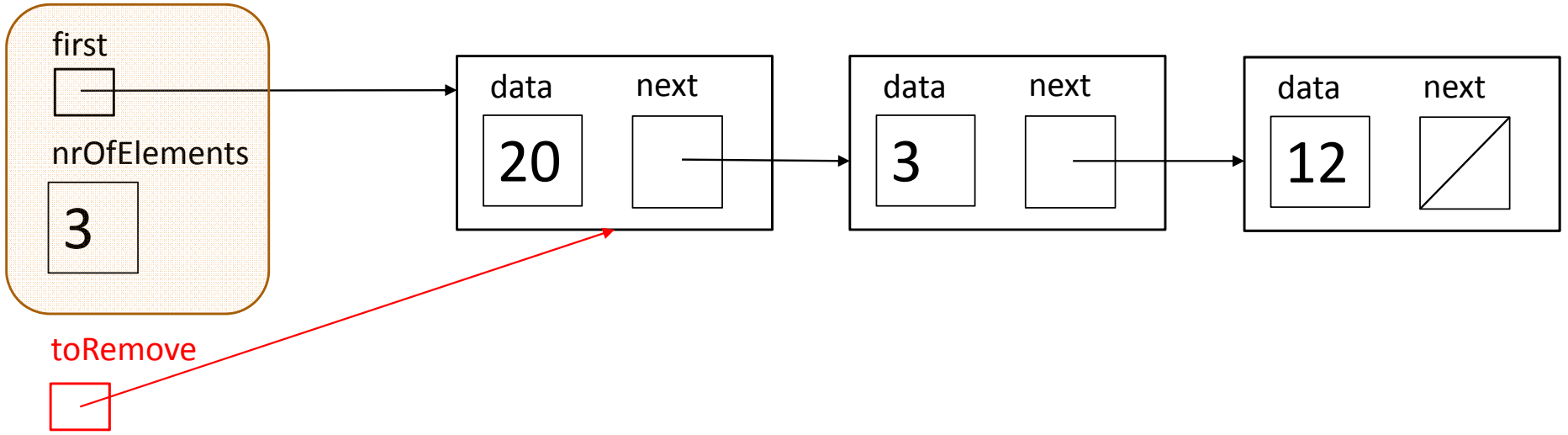
Ta bort första, if (pos == 0)



1. Skapa en ny Node-pekare (som ska användas för att ”peka ut” den nod som ska tas bort) och ge den adressen till första noden
2. Låt första noden bli nuvarande första nodens efterföljare
3. Kopiera innehållet (data-delen) i den nod som ska tas bort
4. Frigör minnet för noden som ska tas bort, minska antalet och returnera kopian av innehållet

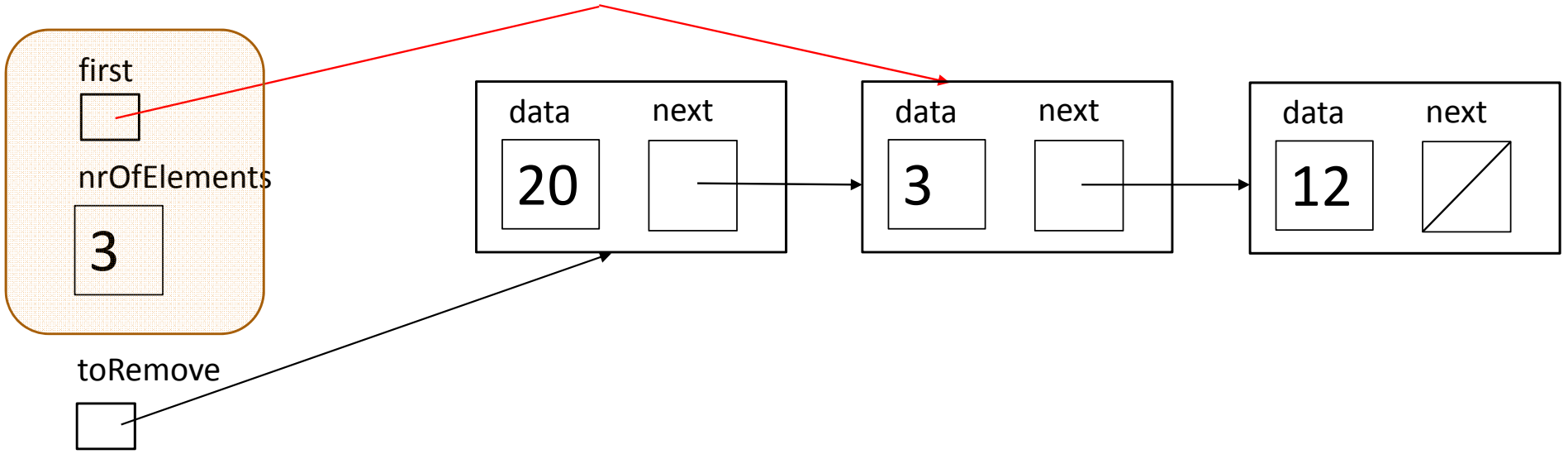
1.Skapa en ny Node-pekare och ge den adressen till första noden

Node* toRemove = this->first;



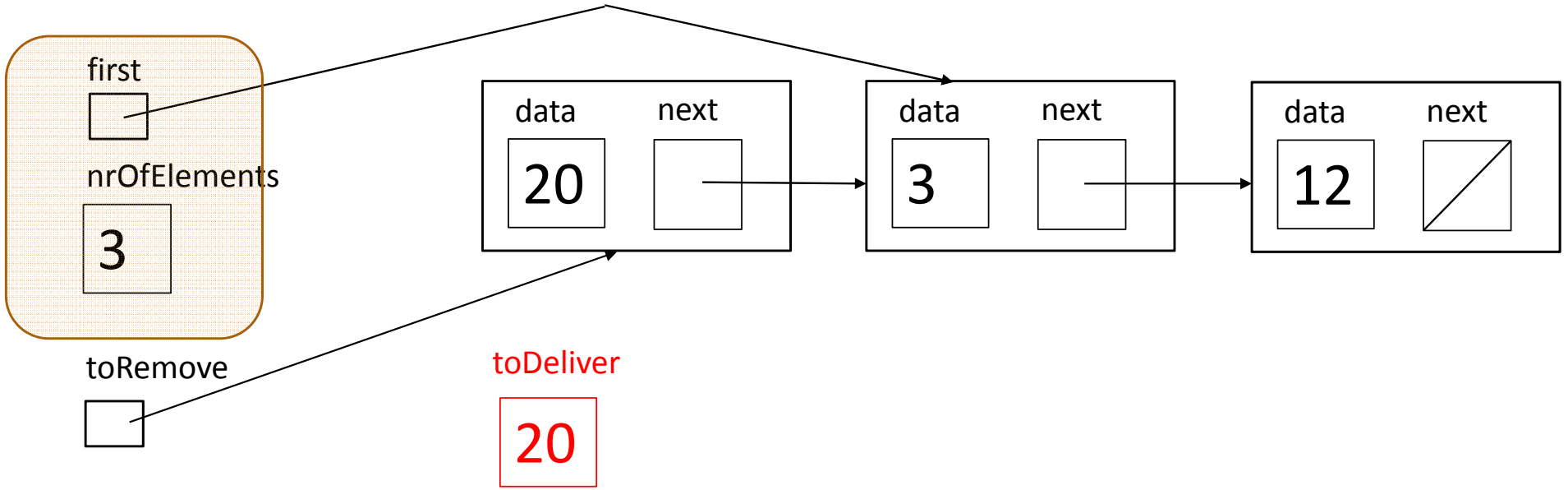
2. Låt första noden bli nuvarande första nodens efterföljare

this->first= this->first->next; eller this->first= toRemove->next;



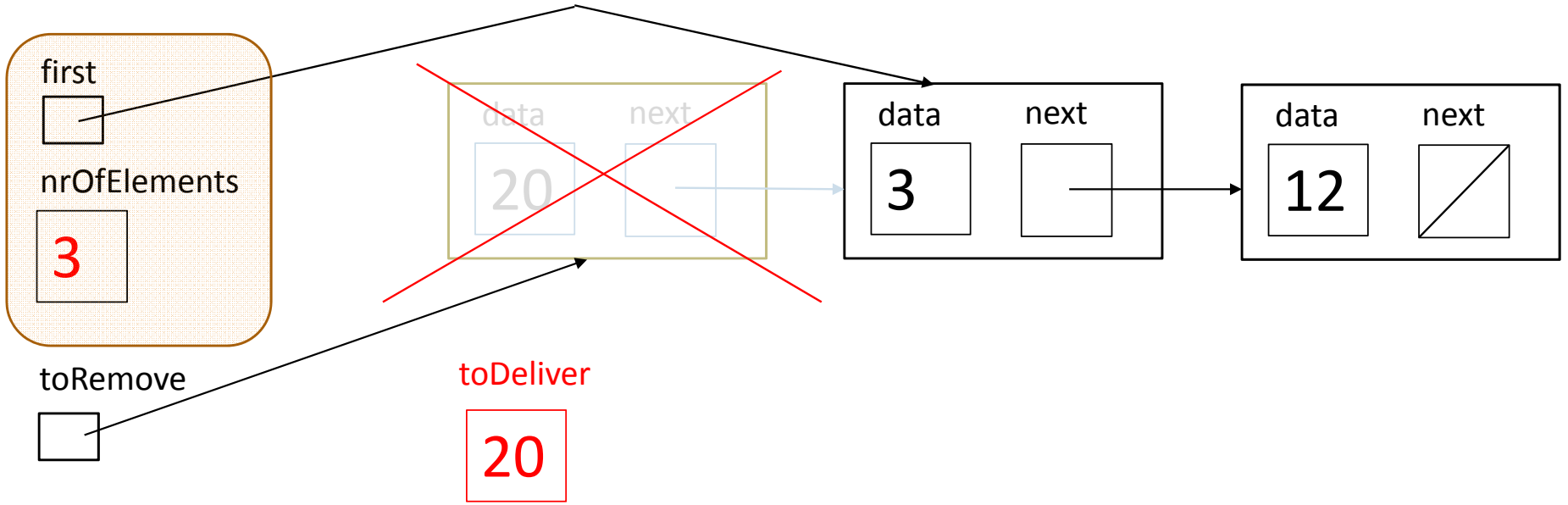
3. Kopiera innehållet (data-delen) i den nod som ska tas bort

toDeliver = toRemove->data;

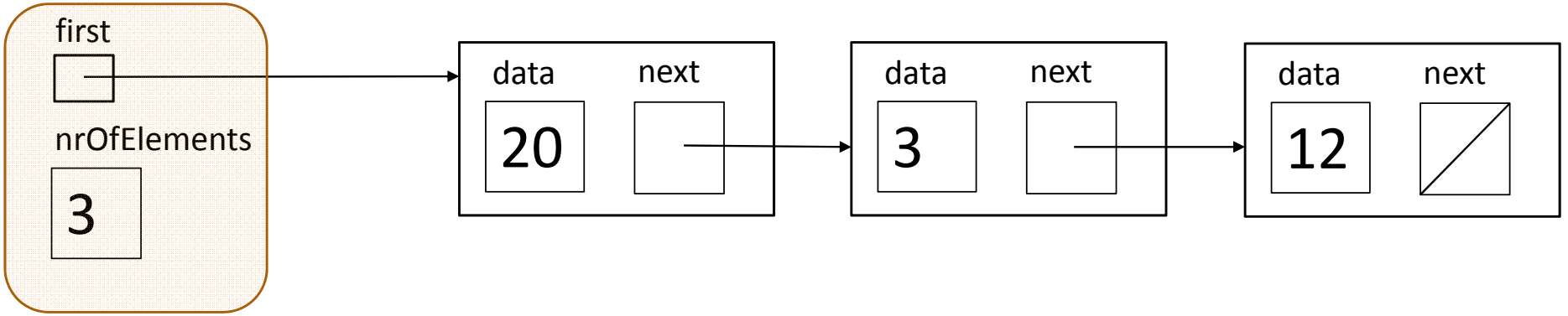


4. Frigör minnet för noden som ska tas bort, räkna ner och returnera kopian av innehållet

delete toRemove ; this->nrOfElements--; return toDeliver;



Ta bort på en position efter 0



1. Skapa en ny Node-pekare, som ska ”peka ut” den nod som ska tas bort och en Node-pekare som ska ”peka ut” noden före den som ska tas bort
2. Vandra med pekarna tills dessa att de pekar ut ”rätt noder”
3. Låt nodens efterföljare till noden före den som ska tas bort bli den nod som r efterföljare till den nod som ska tas bort
4. Kopiera innehållet (data) i de nod som ska tas bort
5. Frigör minnet för noden som ska tas bort och returnera kopian av innehållet

Laboration och inlämning ….

…kommer bl.a. att belysa länkning!!