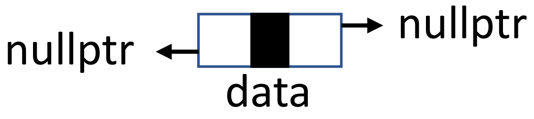
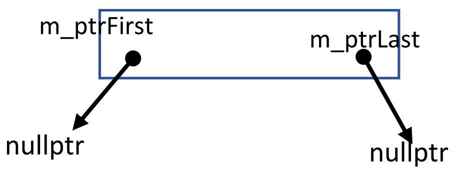
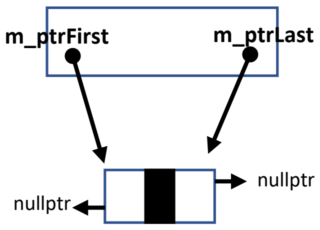
void PushFront(const T& newData)

1. Create a new Node

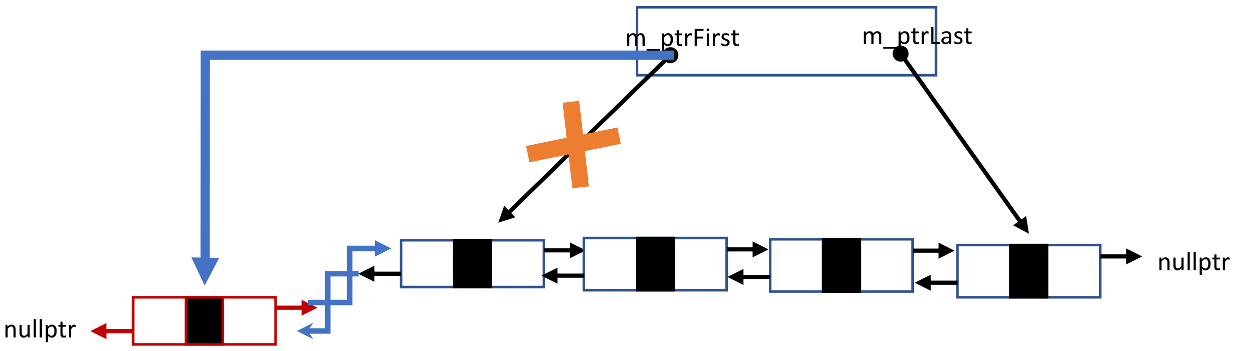


2. If the LinkedList empty





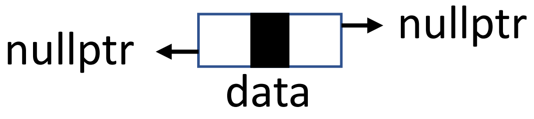
3. If the Linkedlist has at least one element



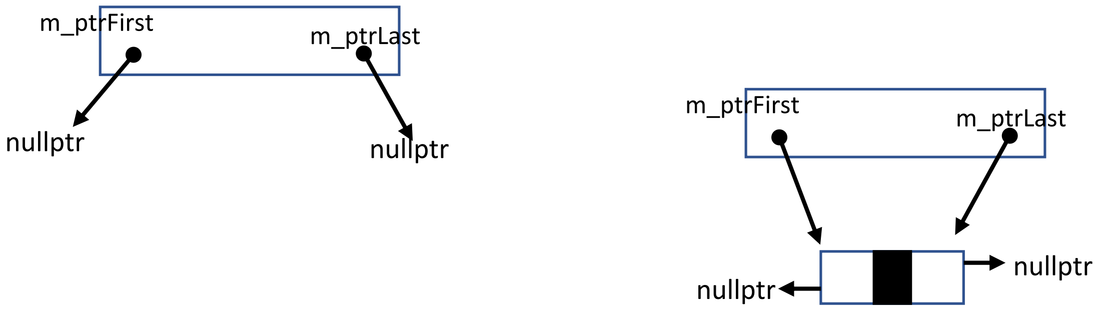
4. m_itemCount++

void PushBack(const T& newData)

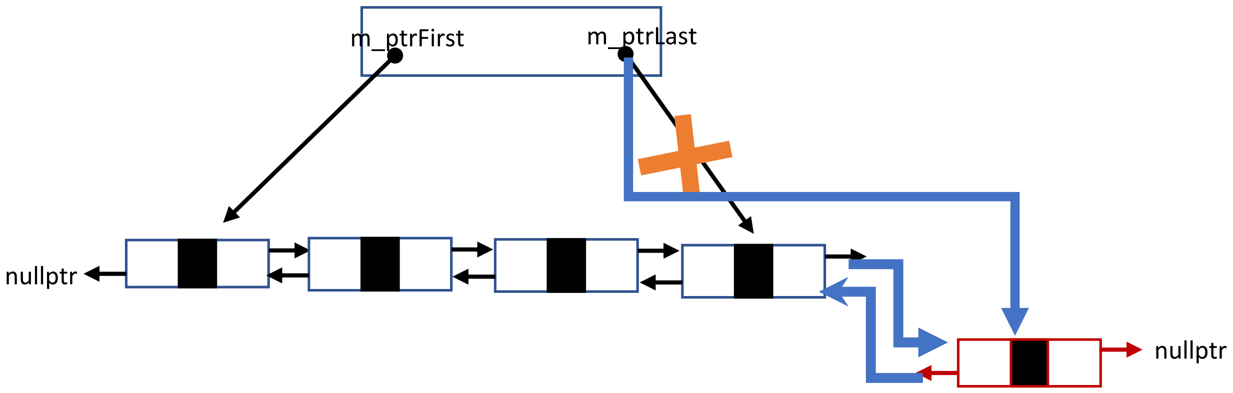
1. Create a new Node



2. If the LinkedList empty



3. If the Linkedlist has at least one element



4. m_itemCount++

void Insert(const T& newData, int atIndex)

1. Invalid index check and throw exception.

2. If the insertion is to the first element (atIndex == 0), call PushFront()

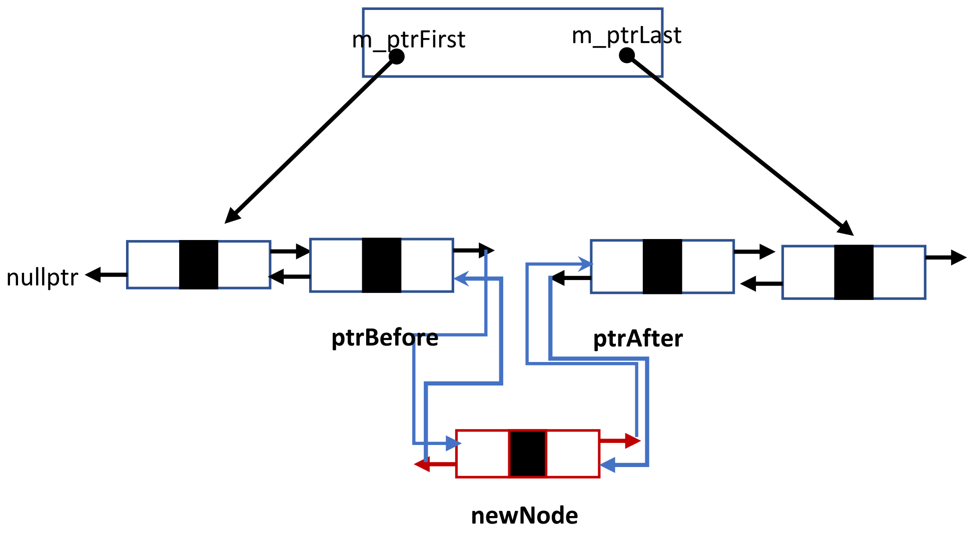
3. Else if the insertion is to the last of list ;
   (atIndex == m_itemCount) then PushBack(NewData)

4. else the insertion is somewhere in the middle;

   define ptrBefore and assign it to the method GetAtIndex(index-1)

   define ptrAfter and assign it to the method GetAtIndex(index)

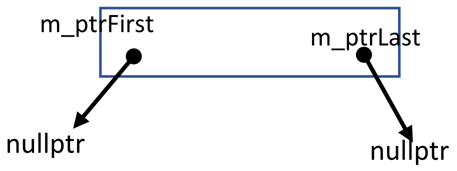
   create a newNode



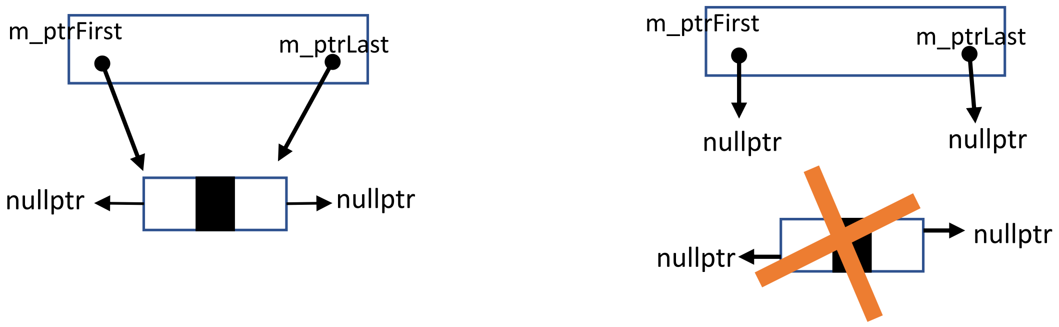
5. m_itemCount++

void PopFront()

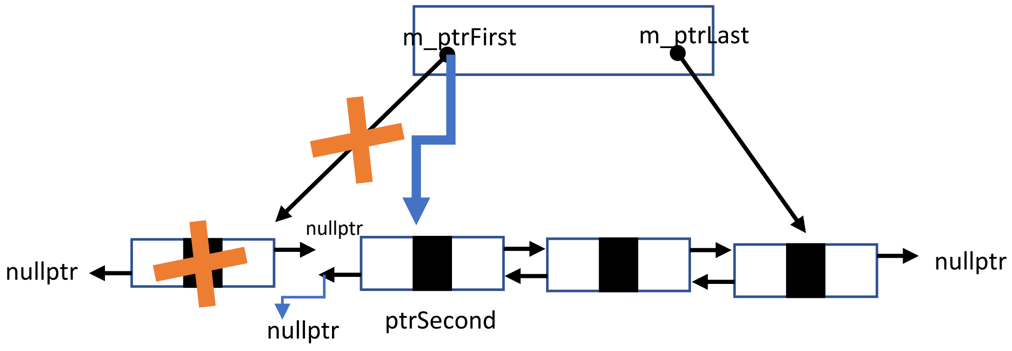
1. If the list is empty, ignore.



2. If the list has only one element;



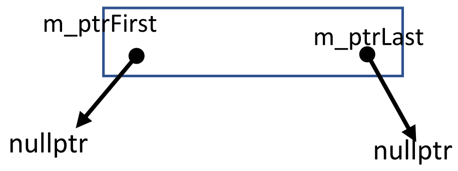
3. If there is more than one element in the list;
   define ptrSecond and set it to m_ptrFirst->ptrNext



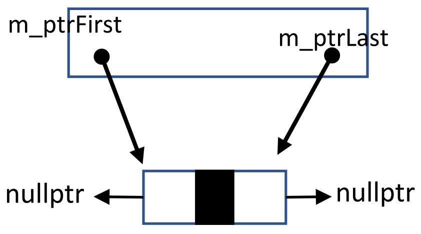
4. m_itemCount --;

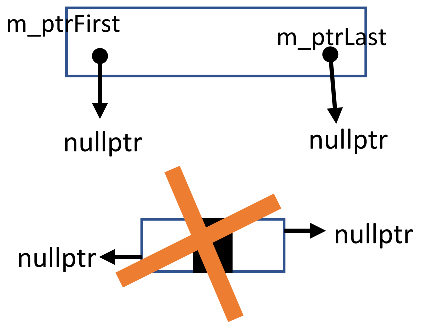
void PopBack()

1. If the list is empty, ignore.

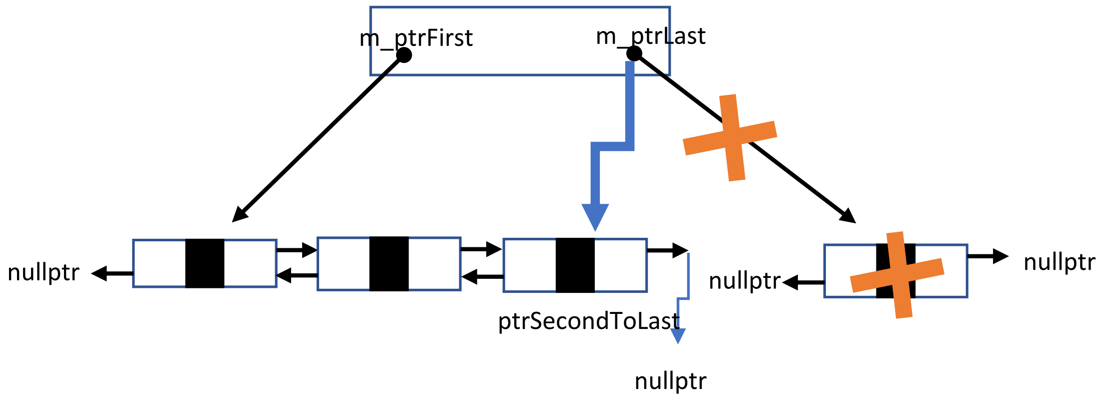


2. If the list has only one element;





3. If there is more than one element in the list;
   define ptrSecondToLast and set it to m_ptrLast->ptrPrev

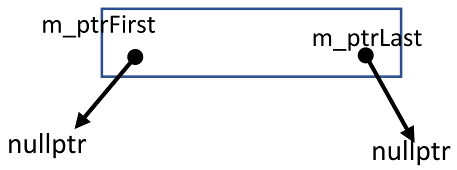


4. m_itemCount --;

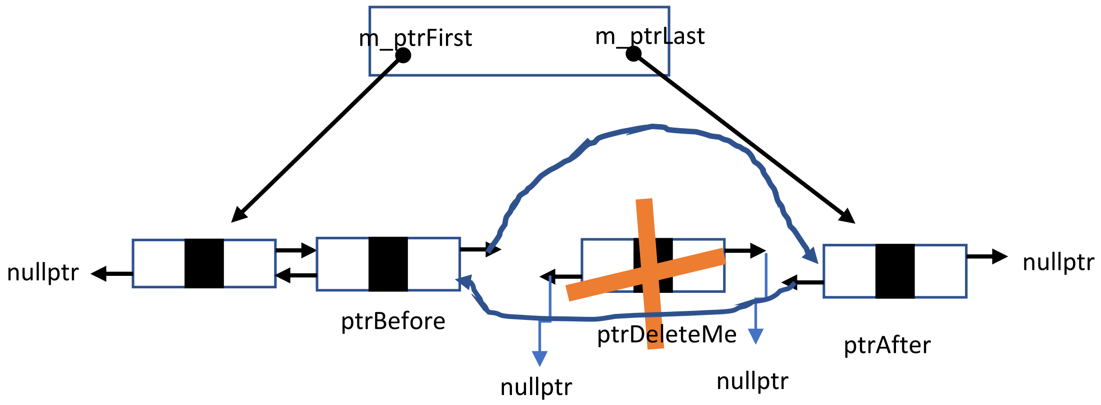
void Remove(int atIndex)

1. Index validation check and throw exception

2. If the list is empty, ignore.



3. If atIndex == 0 then call PushFront() then skip 4
   Else If atIndex == m_itemCount then call PushBack() then skip 4
   else atIndex is in the between;



4. m_itemCount --;