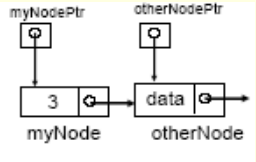
# Lecture 9.2

## Linked List Implementation with Structure



# Node structure

```
struct Node{
    int Data;
    Node * pNode;
};

Node myNode, otherNode;
myNode.Data = 3;
OtherNode.Data = 5;
myNode.pNode = &otherNode;

Node * myNodePtr;
myNodePtr->Data = 3;
myNodePtr->pNode = &otherNode;

Node *otherNodePtr;
myNodePtr->pNode = otherNodePtr;
```






# Linked List Implementation

```
struct NODE {
   NODE *pNext;
   NODE *pPrev;
   int nData;
};

//declare head and tail of list
//originally the list is empty

NODE *pHead = NULL;
NODE *pTail = NULL;
```



# Linked List implementation

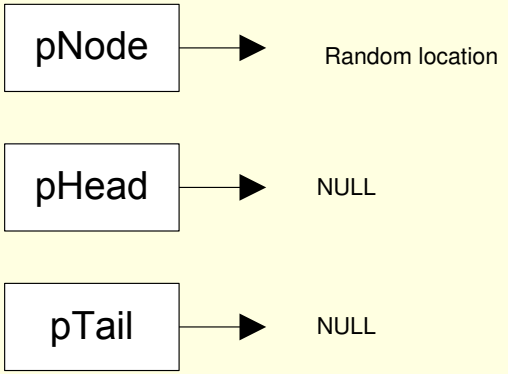
```
void AppendNode ( NODE *pNode);

void InsertNode ( NODE *pNode, NODE *pAfter);

void RemoveNode ( NODE *pNode);

void DeleteAllNodes ( );
```

# LL implementation (cont 2)

```
int main()
{
   NODE *pNode; //declare the pointer for
   // the dynamically allocated memory
```



# LL Implementation (so far)



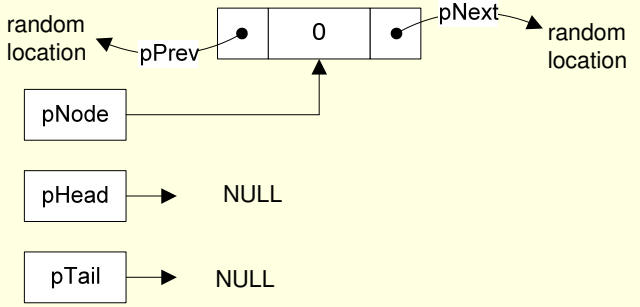



# LL Implementation

```
for (int i = 0; i < 100; i++) {
   pNode = new NODE; //allocate
         // memory for each node and make
         // pointer point to the
         //dynamically allocated memory
   pNode->nData = i; //put some data in the node
   AppendNode(pNode); //add node to list
}
```



# Allocating new node, making pNode point to new node and assigning nData of new node to 0.



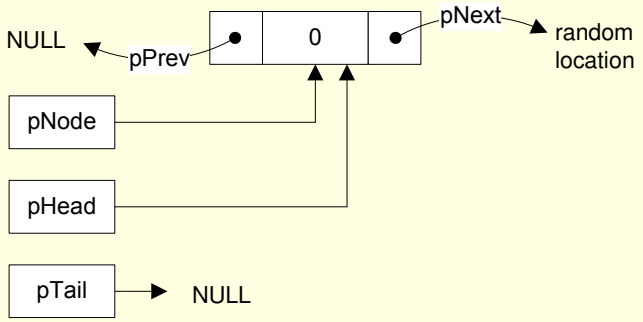
# Appending pNode

```
void AppendNode ( NODE *pNode )
{
    if (pHead == NULL) { //if list is empty
        pHead = pNode; //make head point to pNode
        pNode->pPrev = NULL; }
    else {
        pTail->pNext = pNode; //make tail point to pNode
        pNode->pPrev = pTail; }

    pTail = pNode; //tail is now pNode
    pNode->pNext = NULL; //pNode next now points to NULL
}
```



# Appending a new node

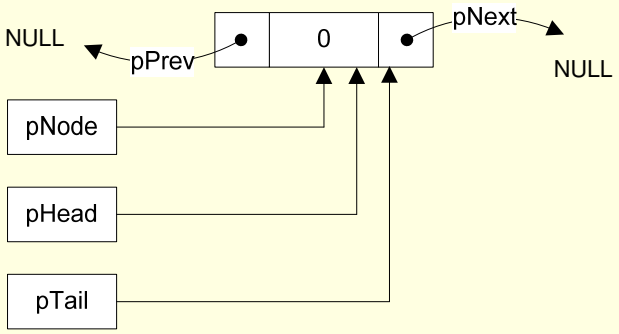
```
if ( pHead == NULL ) {
    pHead = pNode;
    pNode->pPrev = NULL; }
```






# Appending a new node

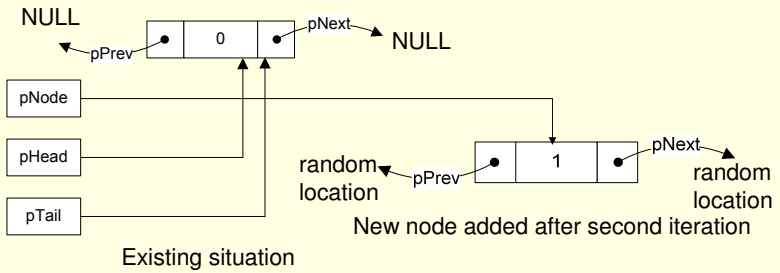
```
pTail = pNode;
pNode->pNext = NULL;
```





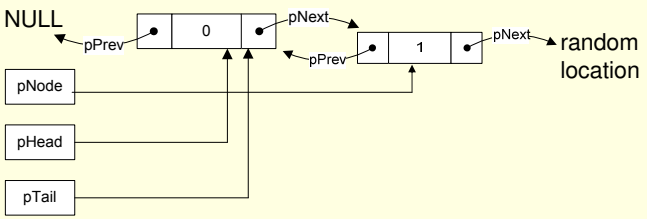

# Second iteration of for loop

- Allocating another new node, making pNode point to new node and assigning nData of new node to 1.



# Appending a new node

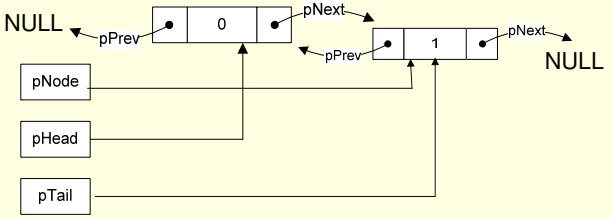
pTail->pNext = pNode; //newly added node
pNode->pPrev = pTail;





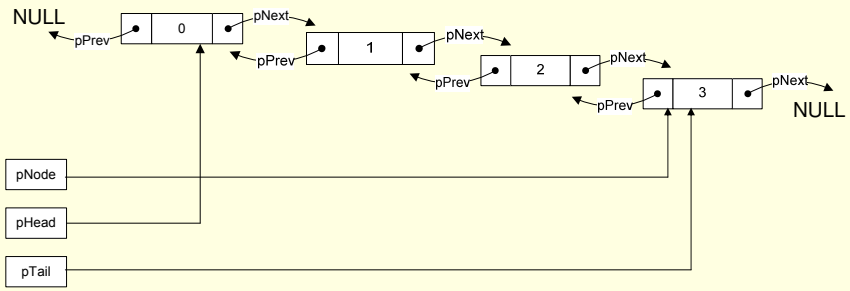

# Appending a new node (cont)

pTail = pNode; //newly added node
pNode->next = NULL;






# After 4 iterations






# Inserting a node

```
void InsertNode(NODE *pNode, NODE *pAfter)
{
   pNode->pNext = pAfter->pNext;
   pNode->pPrev = pAfter;

   if (pAfter->pNext != NULL)
        pAfter->pNext->pPrev = pNode;
   else
        pTail = pNode;

   pAfter->pNext = pNode;
}
```

# Removing Nodes

```
void RemoveNode(NODE *pNode) {
    if (pNode->pPrev == NULL)
        pHead = pNode->pNext;
    else
        pNode->pPrev->pNext = pNode->pNext;

    if (pNode->pNext == NULL)
        pTail = pNode->pPrev;
    else
        pNode->pNext->pPrev = pNode->pPrev;

    delete pNode;
}
```



# Delete all Nodes

```
void DeleteAllNodes()
{
   while (pHead != NULL) //keep on
   //removing until the
   //head points to NULL
   RemoveNode(pHead);
}
```



# Traversing the list

```
for (pNode = pHead; pNode != NULL; pNode = pNode->pNext)
    cout << pNode->nData <<endl;
```