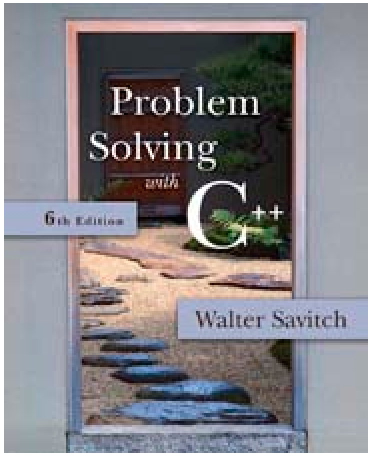
APS105: Lecture 32

Wael Aboelsaadat

wael@cs.toronto.edu

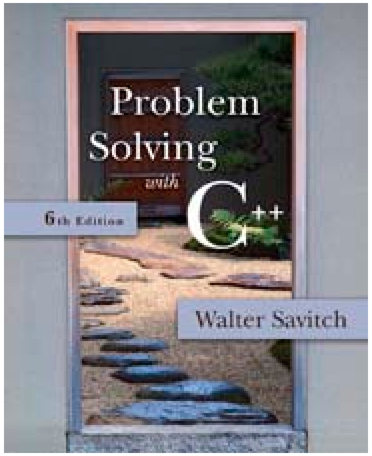
http://ccnet3.utoronto.ca/20079/aps105h1f/

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley



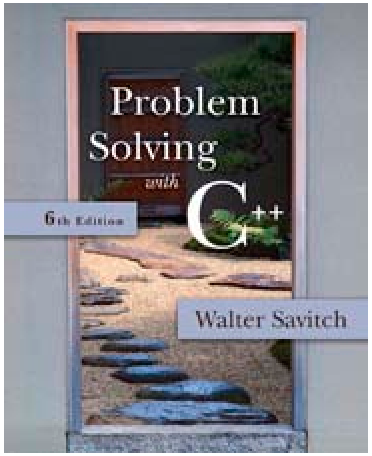
Chapter 13

Pointers and Linked Lists



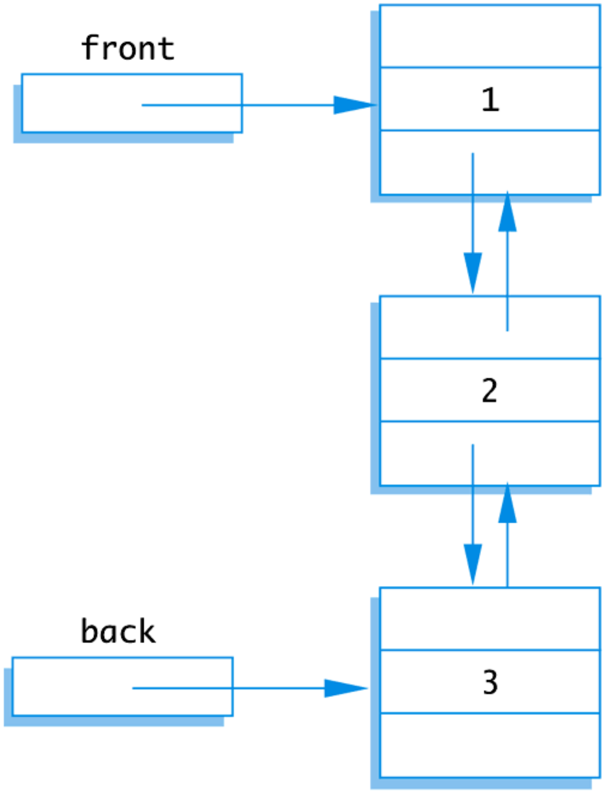
13.1

Nodes and Linked Lists

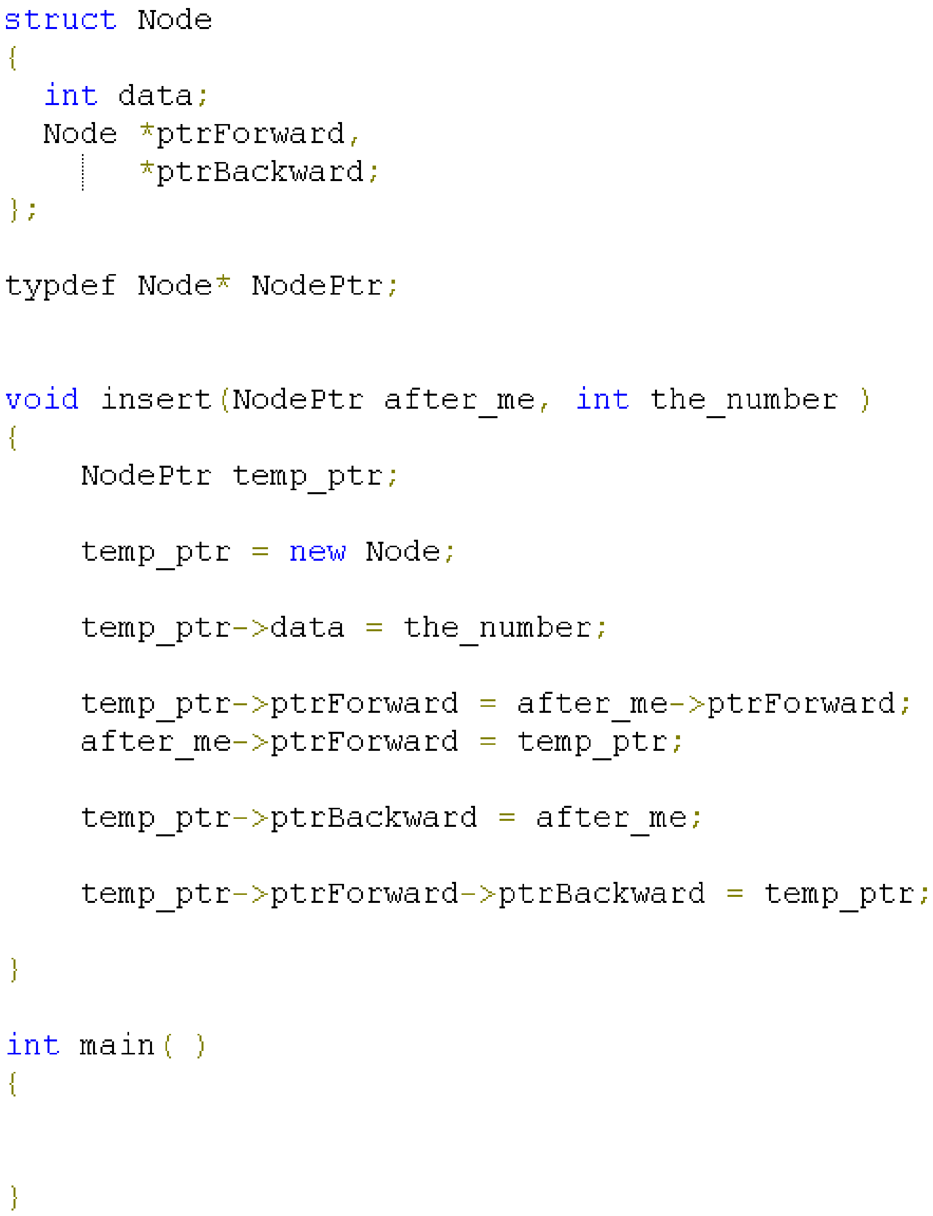


Doubly Linked List

DISPLAY 13.11 A Doubly Linked List



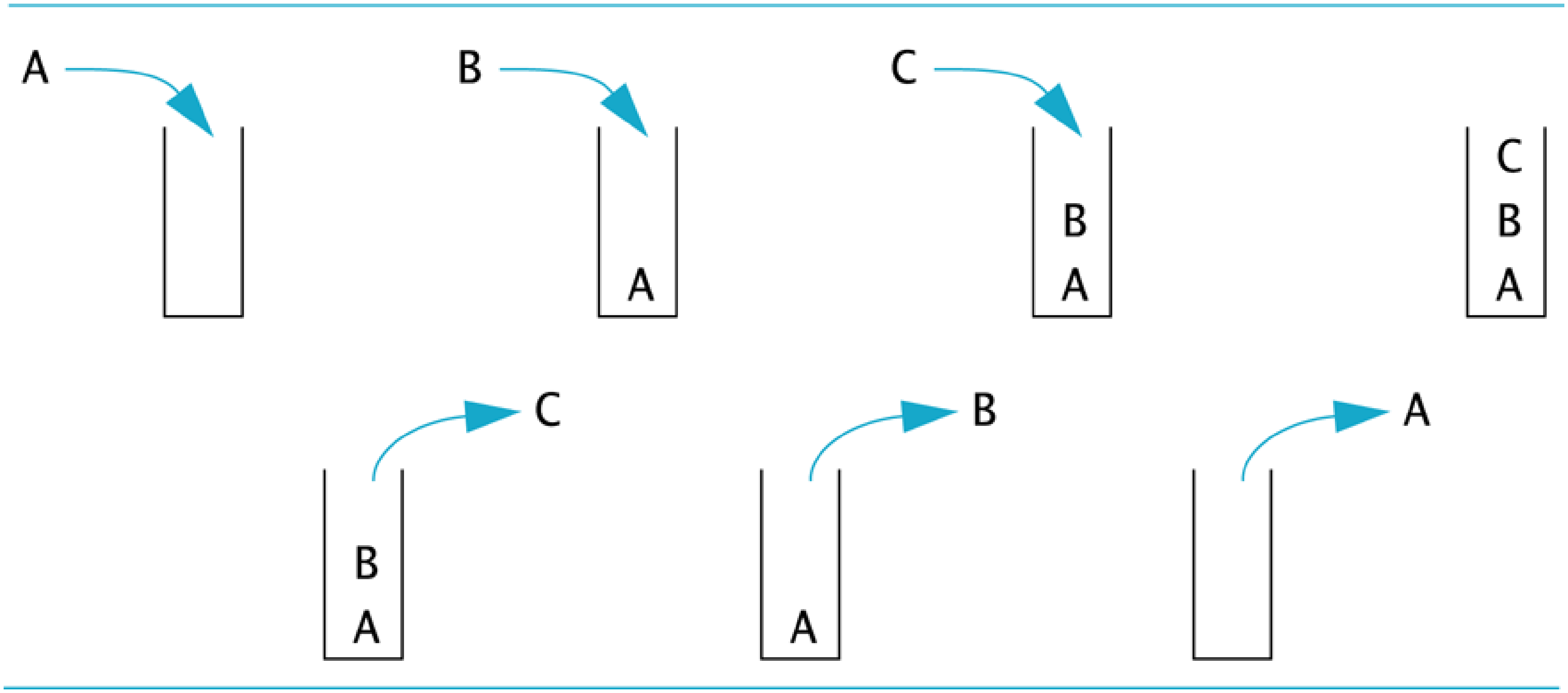
Inserting a New Node into a doubly linked list



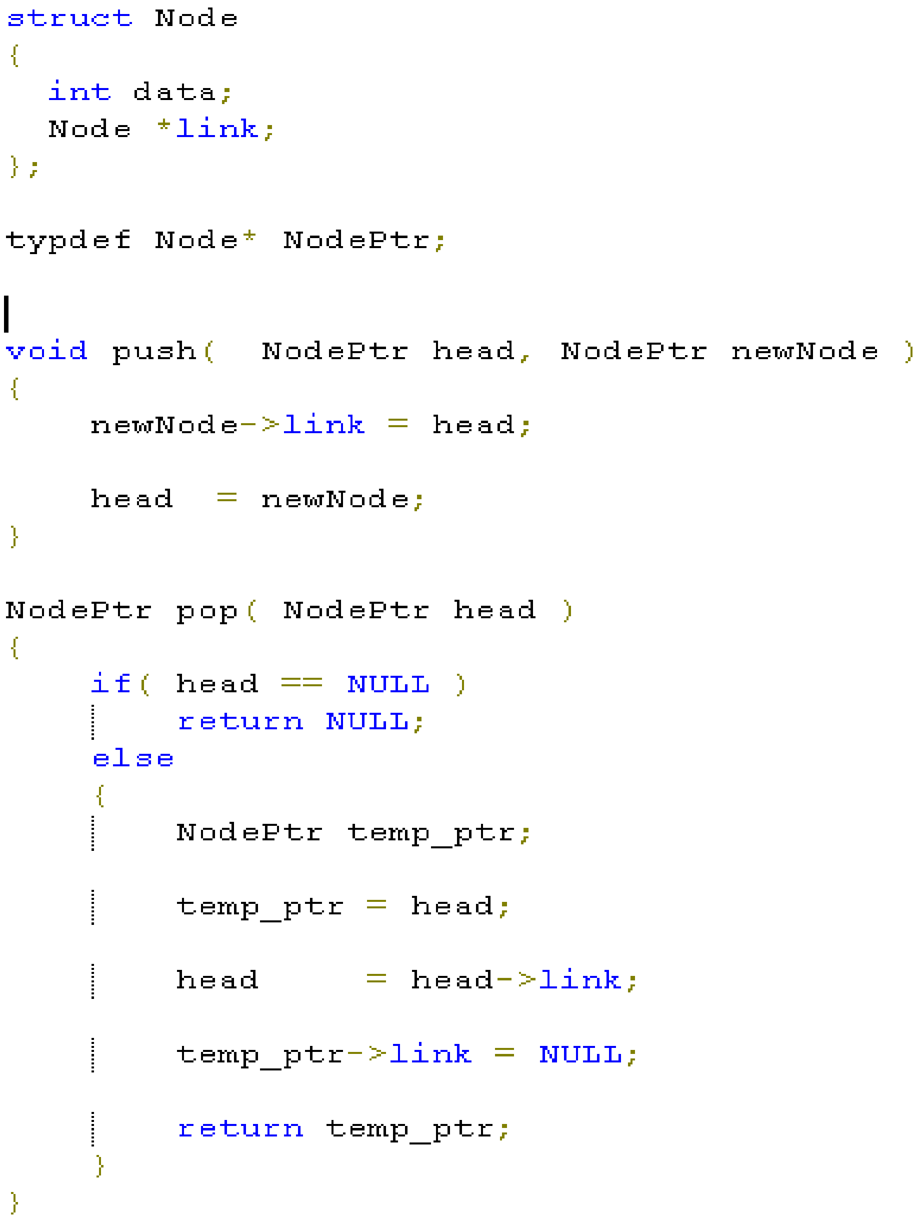
```
struct Node
{
  int data;
  Node *ptrForward,
       *ptrBackward;
};

typdef Node* NodePtr;


void insert(NodePtr after_me, int the_number )
{
    NodePtr temp_ptr;

    temp_ptr = new Node;

    temp_ptr->data = the_number;

    temp_ptr->ptrForward = after_me->ptrForward;
    after_me->ptrForward = temp_ptr;

    temp_ptr->ptrBackward = after_me;

    temp_ptr->ptrForward->ptrBackward = temp_ptr;

}

int main( )
{


}
```

Stack

A Stack

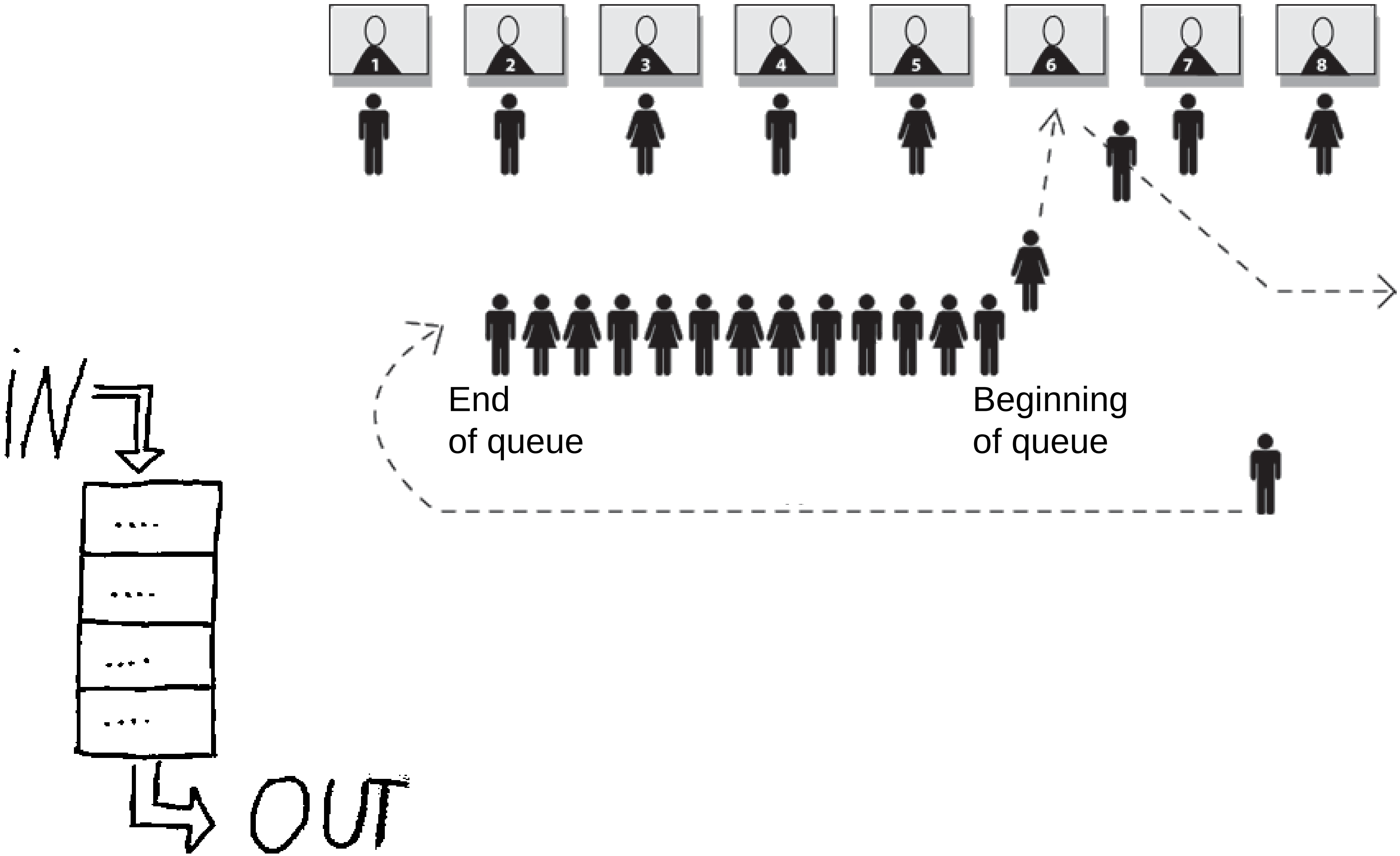


Pushing and Popping element from stack



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

typdef Node* NodePtr;


void push(  NodePtr head, NodePtr newNode )
{
    newNode->link = head;

    head  = newNode;
}

NodePtr pop( NodePtr head )
{
    if( head == NULL )
        return NULL;
    else
    {
        NodePtr temp_ptr;

        temp_ptr = head;

        head     = head->link;

        temp_ptr->link = NULL;

        return temp_ptr;
    }
}
```

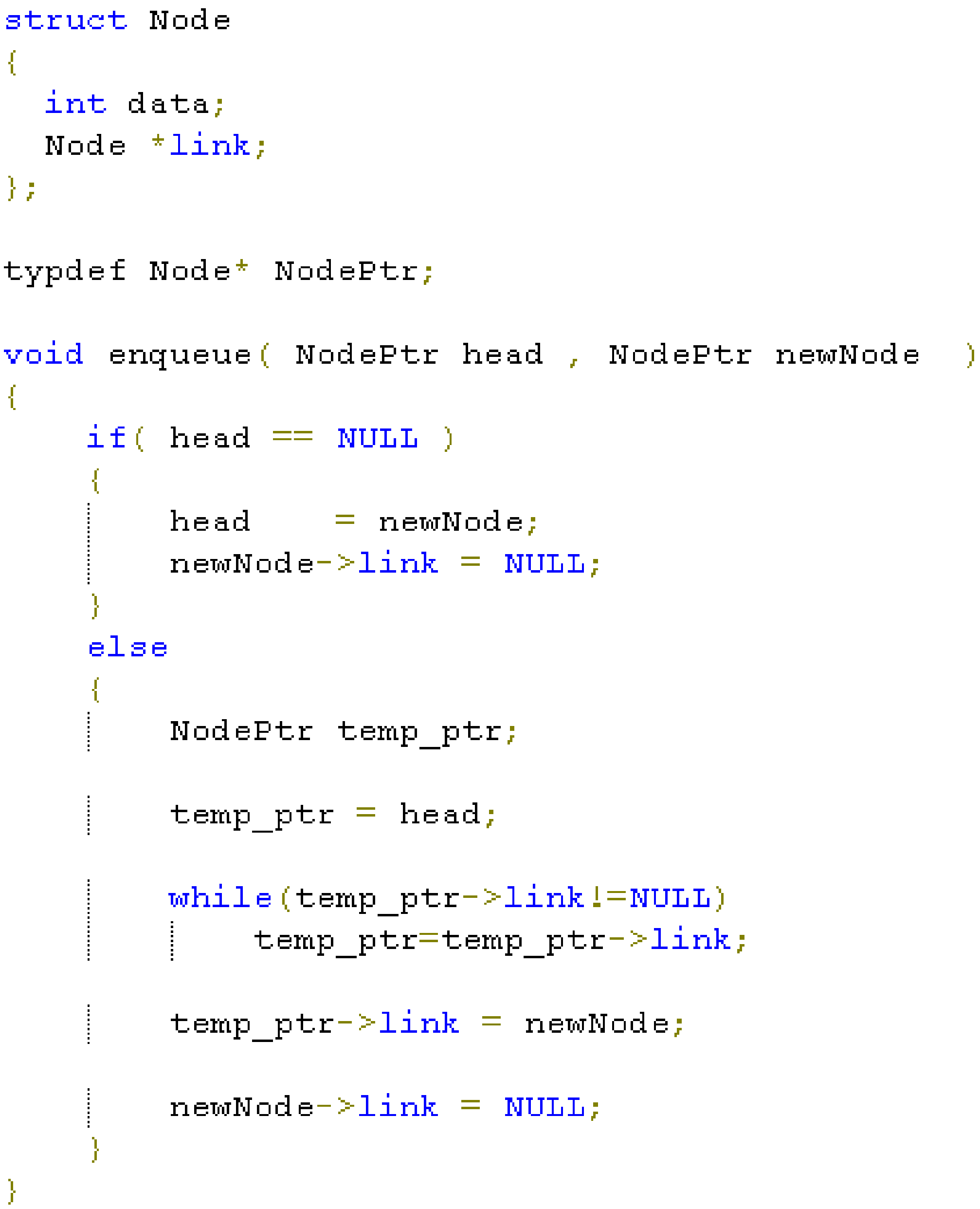
Queue



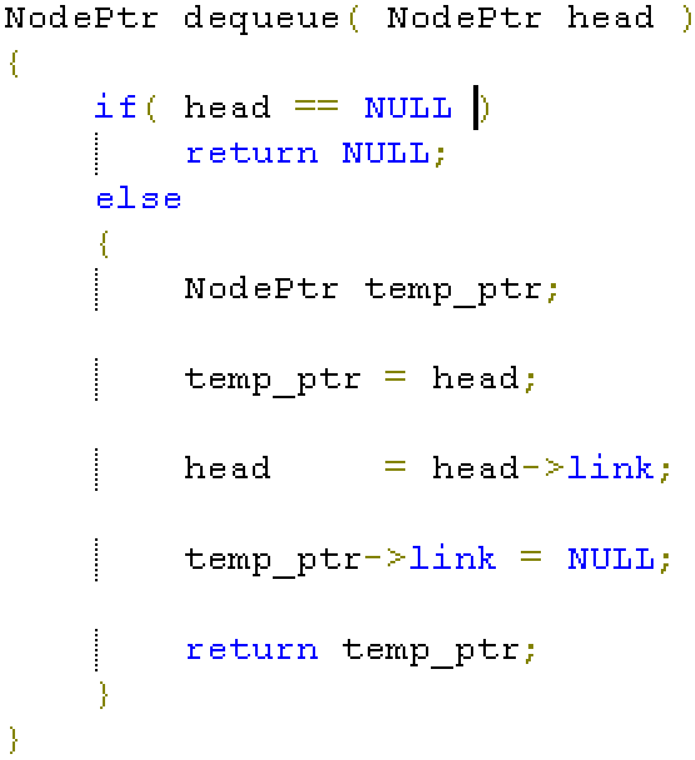
End
of queue

Beginning
of queue

Enqueue and Dequeue elements from a queue



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

typdef Node* NodePtr;

void enqueue( NodePtr head , NodePtr newNode  )
{
    if( head == NULL )
    {
        head    = newNode;
        newNode->link = NULL;
    }
    else
    {
        NodePtr temp_ptr;

        temp_ptr = head;

        while(temp_ptr->link!=NULL)
            temp_ptr=temp_ptr->link;

        temp_ptr->link = newNode;

        newNode->link = NULL;
    }
}
```



```
NodePtr dequeue( NodePtr head )
{
    if( head == NULL )
        return NULL;
    else
    {
        NodePtr temp_ptr;

        temp_ptr = head;

        head     = head->link;

        temp_ptr->link = NULL;

        return temp_ptr;
    }
}
```