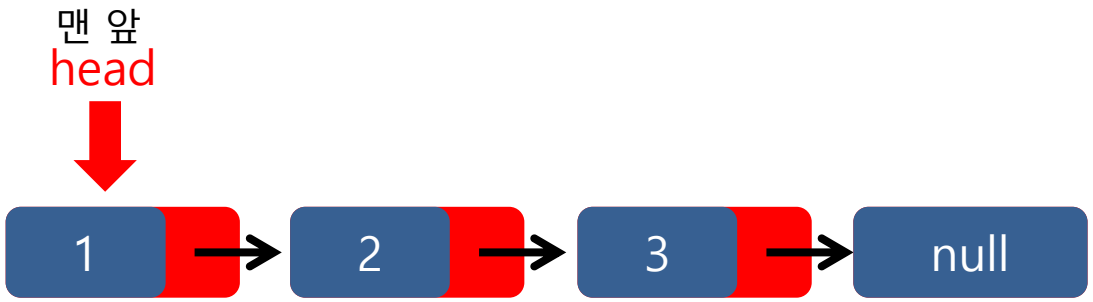
Data structure

Stack, queue

stack

스택은 선입후출

즉 먼저 들어온 데이터가 나중에 나간다!!



멤버변수

```
Node * head;
```

그러므로 탐색이 따로 필요 없습니다.
Head가 가리키는 게 다음 번 반환 데이터!!

stack

멤버함수

```
public:
    Stack() : head(0) {}
    bool IsEmpty();
    void Push(T d);
    T Pop();
    T Peek();
};
```

스택이 비었는가?

Data의 삽입(insert)

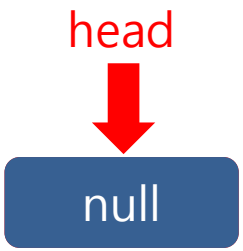
Data의 탐색 및 삭제

Data의 탐색

stack

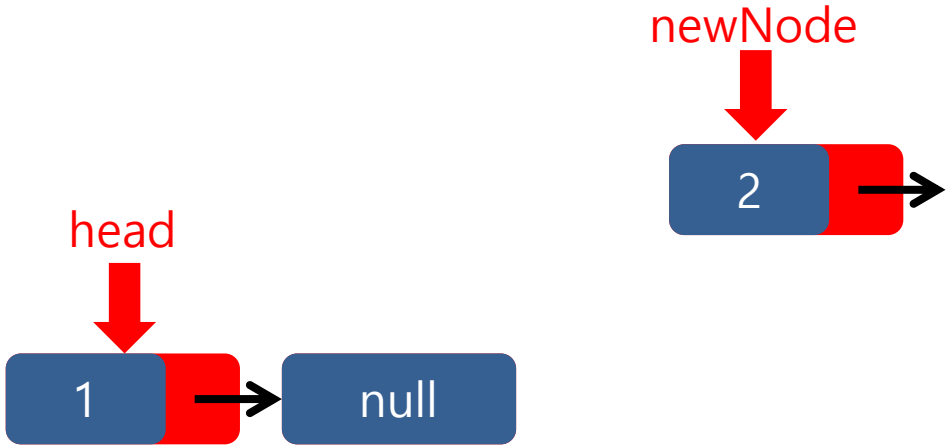
```
bool Stack<T>::IsEmpty()
{
    if (head == 0)
        return true;

    return false;
}
```

Head가 null이면
데이터가 없는 것!



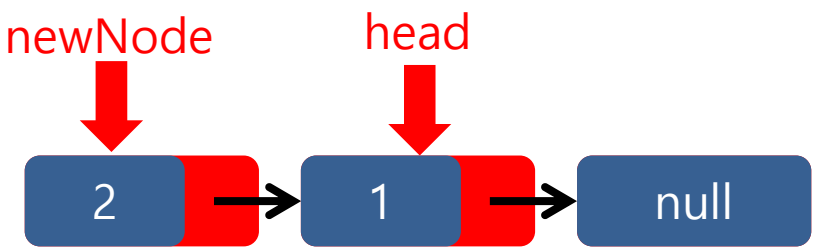
stack

```
void Stack<T>::Push(T d)
{
    Node * newNode = new Node(d);

    newNode->next = head;
    head = newNode;
}
```



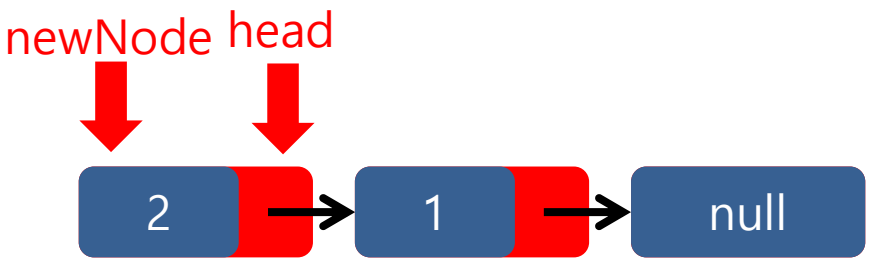
stack

```
void Stack<T>::Push(T d)
{
    Node * newNode = new Node(d);

    newNode->next = head;
    head = newNode;
}
```



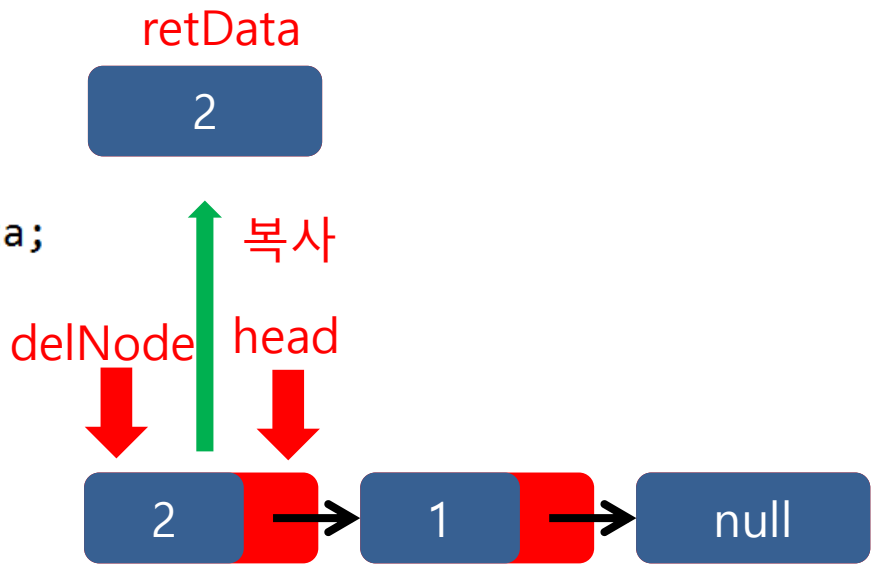
stack

```
void Stack<T>::Push(T d)
{
    Node * newNode = new Node(d);

    newNode->next = head;
    head = newNode;
}
```



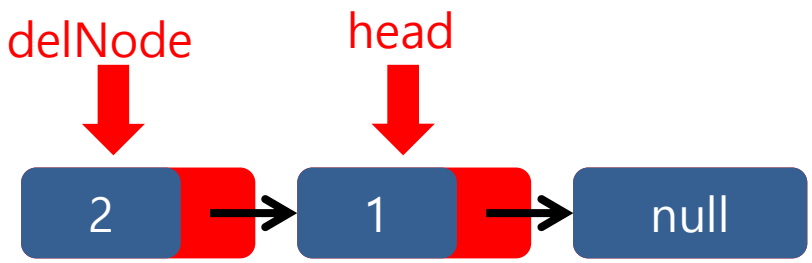
stack

```
T Stack<T>::Pop()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    Node * delNode = head;
    T retData = delNode->data;

    head = head->next;
    delete delNode;
    return retData;
```



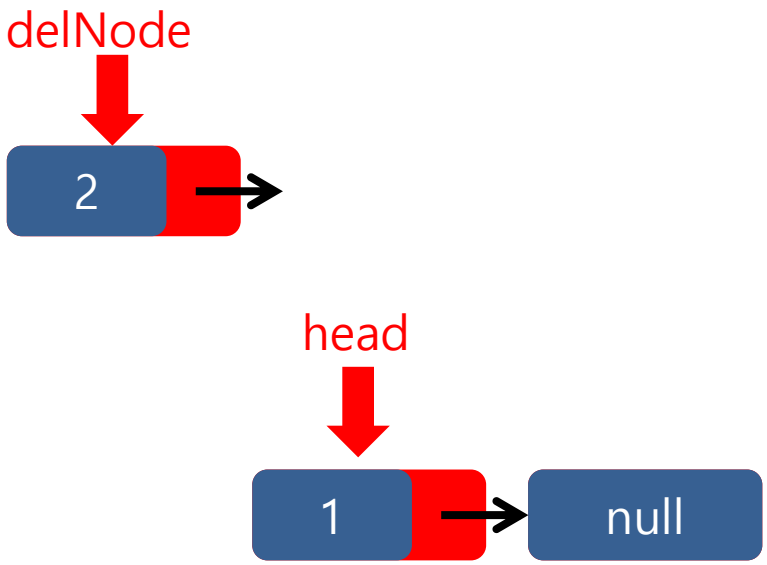
stack

```
T Stack<T>::Pop()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    Node * delNode = head;
    T retData = delNode->data;

    head = head->next;
    delete delNode;
    return retData;
```



stack

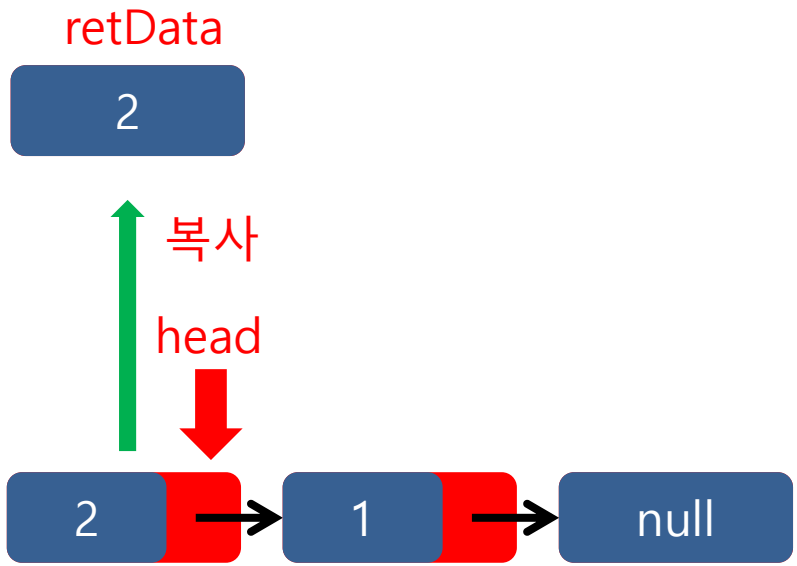
```
T Stack<T>::Pop()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    Node * delNode = head;
    T retData = delNode->data;

    head = head->next;
    delete delNode;
    return retData;
```



stack

```
T Stack<T>::Peek()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    T retData = head->data;
    return retData;
}
```

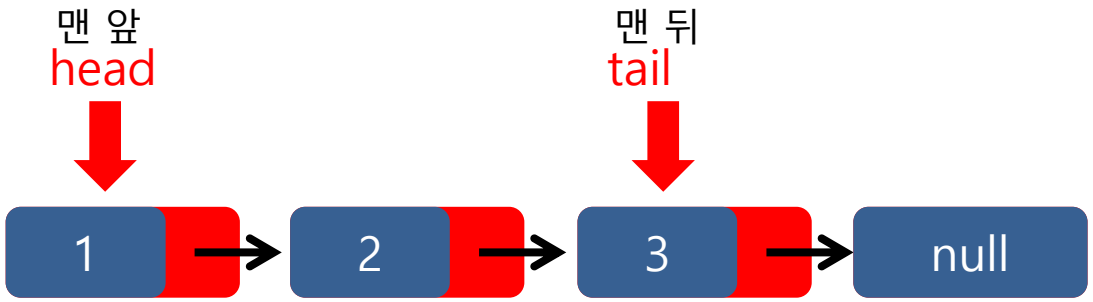
Peek()는 head에 있는 데이터를 반환만 한다



queue

큐은 선입선출

즉 먼저 들어온 데이터가 먼저 나간다!!



멤버변수

```
Node * head;
Node * tail;
```

큐도 역시 탐색이 따로 필요 없습니다.
데이터는 들어오면서 tail에,
Head가 가리키는 게 다음 번 반환 데이터!!

queue

멤버함수

```
Queue() : head(0), tail(0) { }
bool IsEmpty();
void Enqueue(T d);
T Dequeue();
T Peek();
```

스택이 비었는가?

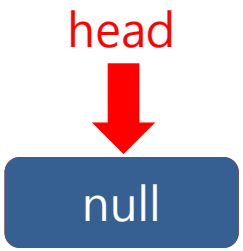
Data의 삽입(insert)

Data의 탐색 및 삭제

Data의 탐색

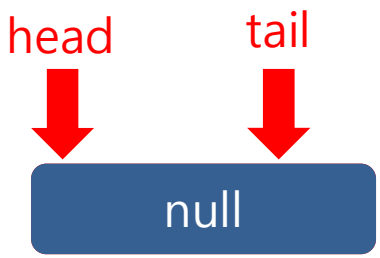
queue

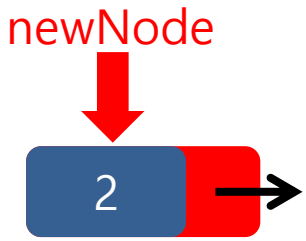
```
bool Queue<T>::IsEmpty()
{
    if (head == 0)
        return true;

    return false;
}
```



queue

```
void Queue<T>::Enqueue(T d)
{
    Node * newNode = new Node(d);

    if (head == 0)
    {
        head = newNode;
        tail = newNode;
        return;
    }

    tail->next = newNode;
    tail = newNode;
}
```

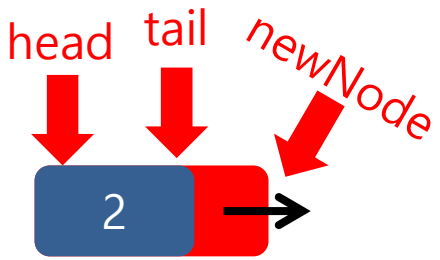




queue

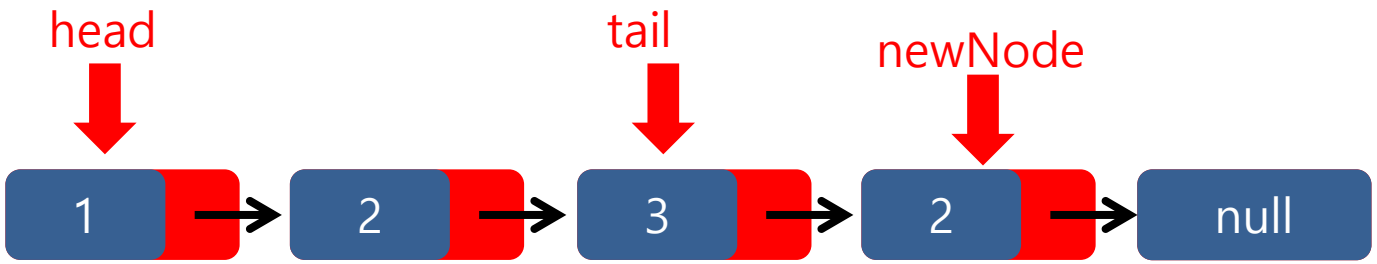
```
void Queue<T>::Enqueue(T d)
{
    Node * newNode = new Node(d);

    if (head == 0)
    {
        head = newNode;
        tail = newNode;
        return;
    }

    tail->next = newNode;
    tail = newNode;
}
```

데이터 최초 insert 시



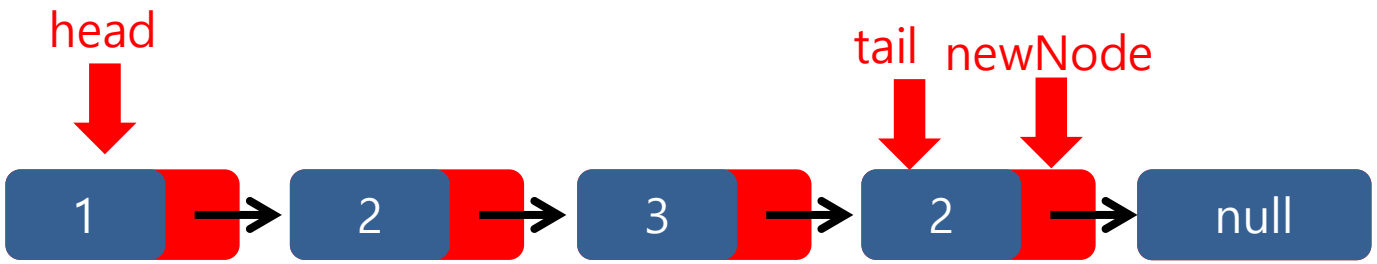
queue

```
void Queue<T>::Enqueue(T d)
{
    Node * newNode = new Node(d);

    if (head == 0)
    {
        head = newNode;
        tail = newNode;
        return;
    }

    tail->next = newNode;
    tail = newNode;
}
```



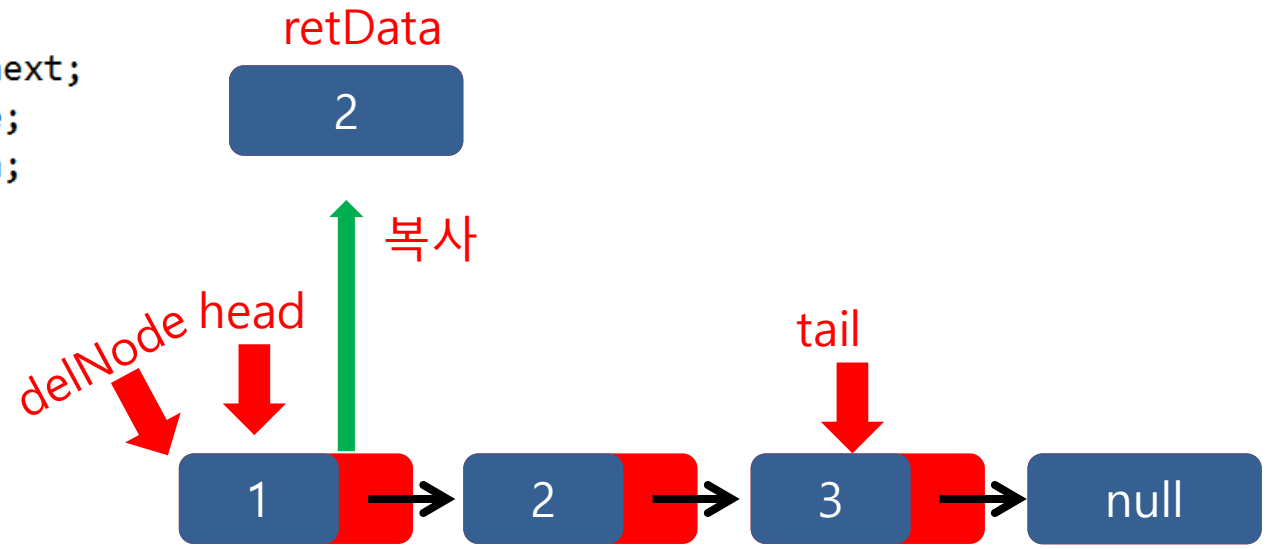
queue

```
void Queue<T>::Enqueue(T d)
{
    Node * newNode = new Node(d);

    if (head == 0)
    {
        head = newNode;
        tail = newNode;
        return;
    }

    tail->next = newNode;
    tail = newNode;
}
```



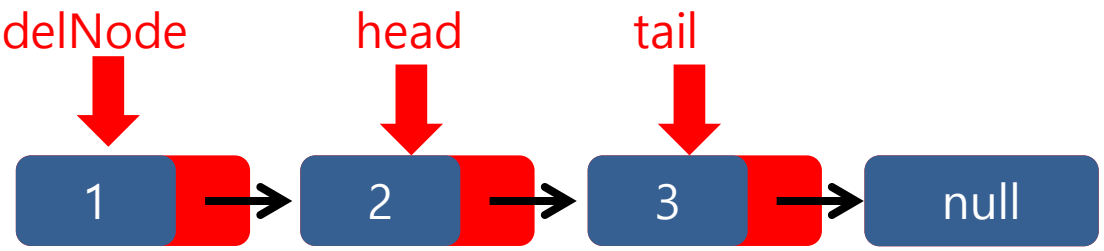
queue

```
T Queue<T>::Dequeue()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    Node * delNode = head;
    T retData = delNode->data;

    head = head->next;
    delete delNode;
    return retData;
}
```



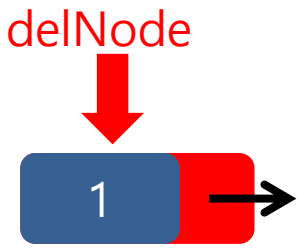
queue

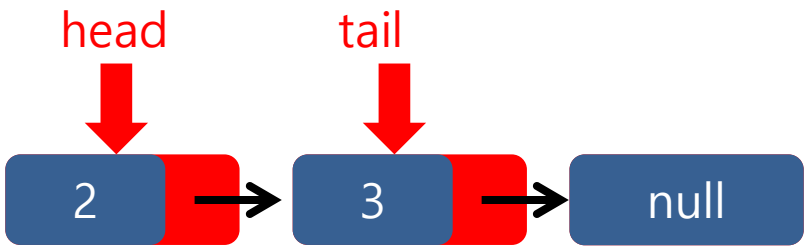
```
T Queue<T>::Dequeue()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    Node * delNode = head;
    T retData = delNode->data;

    head = head->next;
    delete delNode;
    return retData;
}
```



queue

```
T Queue<T>::Dequeue()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    Node * delNode = head;
    T retData = delNode->data;

    head = head->next;
    delete delNode;
    return retData;
}
```





queue

```
T Queue<T>::Peek()
{
    if (IsEmpty())
    {
        cout << "Error!" << endl;
        exit(-1);
    }

    return head->data;
}
```

Peek()는 head에 있는 데이터를 반환만 한다