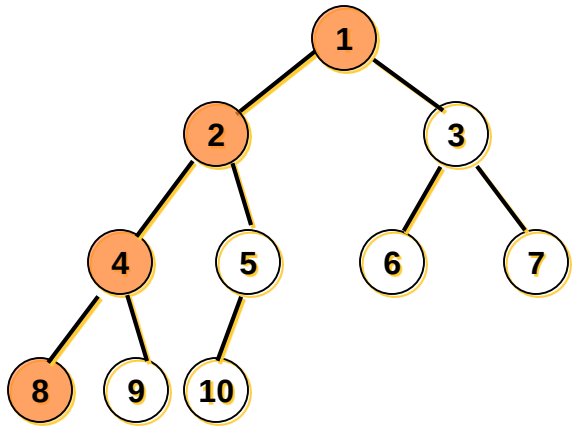
트리 연습 2

- 파일 data.txt에 아래와 같은 정수값들이 저장되어 있다.
  - 1 2 3 4 5 6 7 8 9 10
- 링크드 리스트 이용하여 책의 p 255의 그림 7-18에 있는것과 같이 완전 이진 트리를 만든후에
  - 단 책에 있는 함수를 사용하지 말고 main()함수에서 전부 처리하는 프로그램을 작성하라
- 왼쪽의 그림에 색칠된 노드를 따라가면서 출력하는 프로그램을 작성하라.
  - 포인터를 이용하여 현재 노드의 왼쪽 자식노드를 찾아가는 프로그램임



링크드 리스트를 이용한 트리

- 문제의 의도를 파악하자 (단, 본예제는 이 문제에만 작동할뿐 범용적이지는 않음)
  - 파일 data.txt에 정수값들이 저장되어 있다.
    - 파일을 열어야 한다. FILE포인터를 사용하자
  - 메인함수 내에서 처리해야 한다
    - 재귀함수 호출은 사용하지 않는다
  - 링크드 리스트에 이 값들을 저장하고 완전 이진트리를 만들자~!!
    - 트리는 오른쪽 링크와 왼쪽 링크를 가르키고 있어야 한다
    - 오른쪽 링크와 왼쪽 링크를 저장할 수 있고 정수를 저장하는 구조체를 쓰자~!!
  - 트리의 왼쪽 링크만 따라가면서 정수를 출력하는 프로그램을 만들자

링크드 리스트 실습 Solution

- 일단 오른쪽 노드와 왼쪽 노드를 기억할 수 있고 정수형 데이터를 저장할 수 있는 구조체를 만들자

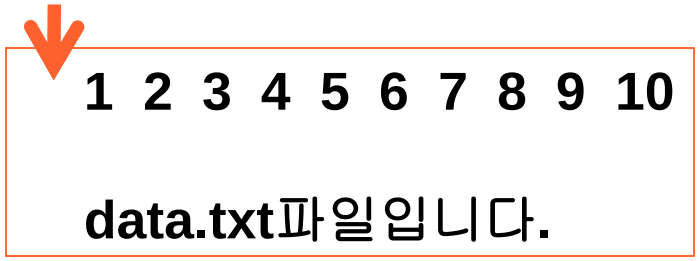
```
// 구조체 정의
typedef struct TreeNode
{
    struct TreeNode *pLeft;          // 트리의 왼쪽을 기억할 노드 포인터
    int data;                        // 실제 Data값
    struct TreeNode *pRight;         // 트리의 오른쪽을 기억할 노드 포인터
}TreeNode;
```

링크드 리스트 실습 Solution

- 파일을 읽어 들이자

```
// 1. 파일포인터를 통하여 파일을 연다
FILE* pFile = fopen("data.txt", "r");        // File포인터 선언 및 초기화

// 방어코드 파일이 열리지 않았을 경우
if ( pFile == NULL)
{
    printf("File Open Fail~!!!!");           // error Message
    exit(0);                                 // 프로그램 종료
}
```

pFile



링크드 리스트 실습 Solution

- 단일 연결리스트와 마찬가지로 처음을 기억하고 있어야 링크의 다음을 찾을 수 있다.
  - 트리의 경우 최상위 노드만 기억하면 된다

```
TreeNode *pRoot;     // 처음을 기억하기 위한 부분
TreeNode *pTemp;     // 구조체 할당을 위한 임시 트리노드 포인터변수
pTemp = (TreeNode*)malloc(sizeof(TreeNode));
if ( fscanf( pFile, "%d", &pTemp->data ) )
{
    pTemp->pLeft  = NULL;
    pTemp->pRight = NULL;
    pRoot = pTemp;   // 처음을 기억하기 위한 부분
}
```

pTemp
pRoot

pFile

pTemp->data = 1

1 2 3 4 5 6 7 8 9 10

data.txt파일입니다.

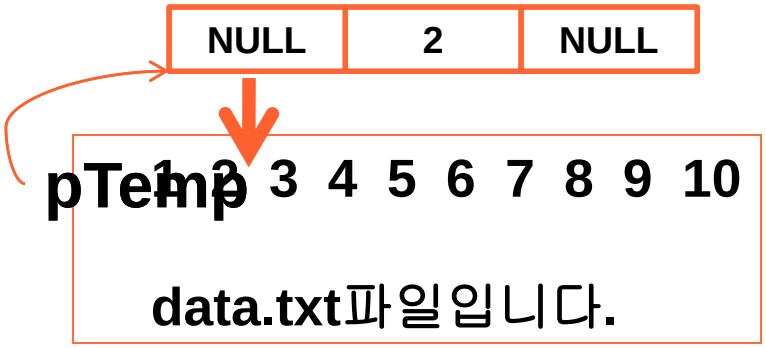
링크드 리스트 실습 Solution

- 파일의 끝까지 읽어 읽어 들이자
  - 파일을 읽으면서 노드 생성
  - 생성된 노드를 완전 이진 트리를 위해 왼쪽 부터 채워준다
  - 재귀함수 호출을 못한다
    - if와 else문으로 구분해야 한다.

링크드 리스트 실습 Solution

- 파일을 읽어 들이는 코드

```
// 파일의 끝까지 돌면서 노드를 생성하고 트리를 완성시킨다
while ( !feof(pFile) )
{
    int nTemp;                                          // 파일의 숫자를 받아오기위한 변수
    TreeNode *pTemp;                                    // 구조체 할당을 위한 임시 트리노드 포인터변수
    TreeNode *pCurrent;                                 // 부모를 알기위한 트리노드 포인터 변수

    if ( fscanf( pFile, "%d", &nTemp ) )                // fscanf읽기 성공일때 0외의 값, 읽기 실패일때 0
    {
        // 새로운 트리노드의 생성
        pTemp = (TreeNode*)malloc( sizeof(TreeNode) );  // 트리노드 사이즈만큼 동적할당
        pTemp->data     = nTemp;                        // 새로만들어진 트리노드의 data에 파일의 숫자를 입력
        pTemp->pLeft    = NULL;                         // 새로만들어진 트리노드의 왼쪽 포인터값 초기화
        pTemp->pRight   = NULL;                         // 새로만들어진 트리노드의 오른쪽 포인터값 초기화
    }
```



링크드 리스트 실습 Solution

- 구 조

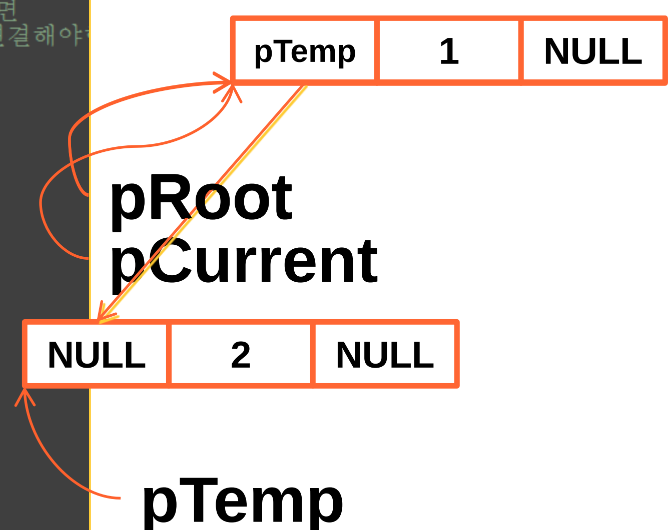
```
while(파일읽기)
{
    if(파일읽기 성공)
    {
        pTemp =새로운 노드 생성
    }
    while(최상위 루트부터 탐색)
    {
        if (비어있는 노드 찾았을 때)
            while 탐색 빠져나감
        else
        {
            if(현재 노드에서 왼쪽 노드의 왼쪽과 오른쪽이 비어있는지 확인)
                현재 노드를 왼쪽 노드로 이동
            else if(현재 노드에서 오른쪽 노드의 왼쪽과 오른쪽이 비어있는지 확인)
                현재 노드를 오른쪽 노드로 이동
            else
                현재 노드를 왼쪽 노드로 이동
        }
    }
    if문을 통해 오른쪽인지 왼쪽인지 결정 후 노드 연결
}
```

다음 **ppt**는 이부분을 나타낸 것이다

링크드 리스트 실습 Solution

```
// 새로운 노드생성 부분 {.....}

pCurrent = pRoot;                          // 부모는 root부터 시작한다

// 부모가 누가 될것인지 알아내기 위해 순환
while ( pCurrent != NULL )
{
    if ( pCurrent->pLeft == NULL ||
         pCurrent->pRight == NULL )        // 현재 노드의 오른쪽을 가르키는 포인터나
    {                                      // 왼쪽을 가르키는 포인터가 비어있다면
        break;                             // 현재 노드에 새로만들어진 노드를 연결해야
    }

    // 현재 노드의 오른쪽과 왼쪽의 노드들이 모두 값을 갖고 있는 경우
    else
    {
        // 현재 노드의 왼쪽 노드의 왼쪽이 비어있거나 오른쪽 노드가 비어있다면
        if ( pCurrent->pLeft->pLeft == NULL ||
             pCurrent->pLeft->pRight == NULL )
        {
            pCurrent = pCurrent->pLeft;    // 현재 노드를 왼쪽 노드로 옮긴다
        }
        // 현재 노드의 오른쪽 노드의 왼쪽이 비어있거나 오른쪽 노드가 비어있다면
        else if ( pCurrent->pRight->pLeft == NULL ||
                  pCurrent->pRight->pRight == NULL )
        {
            pCurrent = pCurrent->pRight;   // 현재 노드를 오른쪽 노드로 옮긴다
        }

        // 현재 노드의 왼쪽 노드와 오른쪽 노드의 왼쪽, 오른쪽 양방향이 다 차있다면
        else
            pCurrent = pCurrent->pLeft;    //완전트리를 위해 왼쪽 노드로 이동한다
    }
}
if ( pCurrent->pLeft == NULL )             // 현제 노드의 왼쪽이 비어있다면
{
    pCurrent->pLeft = pTemp;               // 새로만들어진 노드를 왼쪽에 연결시킨다
}
else if( pCurrent->pRight == NULL )        // 현제 노드의 오른쪽이 비어있다면
{
    pCurrent->pRight = pTemp;              // 새로만들어진 노드를 오른쪽에 연결시킨다
}
```

주어야 함으로 while문 통과



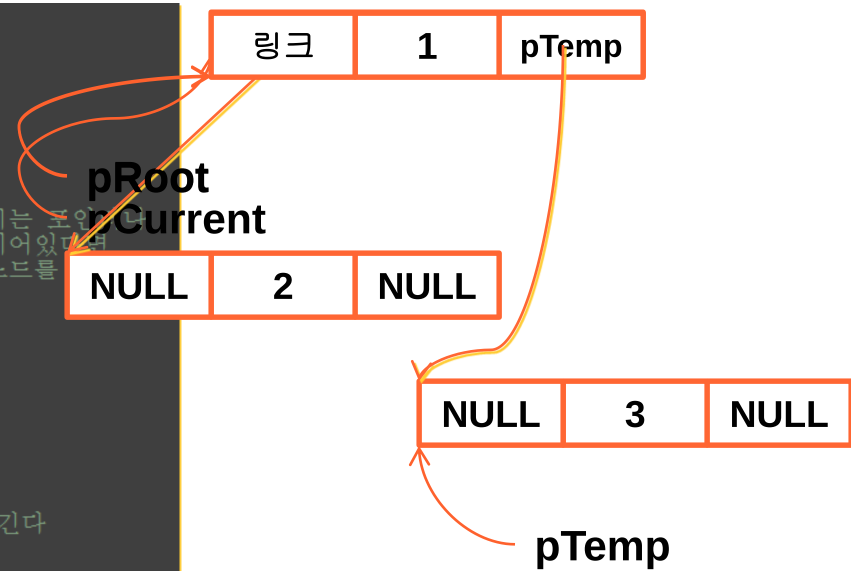
링크드 리스트 실습 Solution

- 파일을 읽어 들이는 코드

```
// 파일의 끝까지 돌면서 노드를 생성하고 트리를 완성시킨다
while ( !feof(pFile) )
{
    int nTemp;                                          // 파일의 숫자를 받아오기위한 변수
                                                        // 구조체 할당을 위한 임시 트리노드 포인터변수
    TreeNode *pCurrent;                                 // 부모를 알기위한 트리노드 포인터 변수

    if ( fscanf( pFile, "%d", &nTemp ) )                // fscanf읽기 성공일때 0외의 값, 읽기 실패일때 0
    {
        // 새로운 트리노드의 생성
        pTemp = (TreeNode*)malloc( sizeof(TreeNode) );  // 트리노드 사이즈만큼 동적할당
        pTemp->data     = nTemp;                        // 새로만들어진 트리노드의 data에 파일의 숫자를 입력
        pTemp->pLeft    = NULL;                         // 새로만들어진 트리노드의 왼쪽 포인터값 초기화
        pTemp->pRight   = NULL;                         // 새로만들어진 트리노드의 오른쪽 포인터값 초기화
    }
```

링크드 리스트 실습 Solution

```
// 새로운 노드생성 부분 {.....}

pCurrent = pRoot;                         // 부모는 root부터 시작한다

// 부모가 누가 될것인지 알아내기 위해 순환
while ( pCurrent != NULL )
{
    if ( pCurrent->pLeft == NULL ||
         pCurrent->pRight == NULL )       // 현재 노드의 오른쪽을 가르키는 포인
    {                                     // 왼쪽을 가르키는 포인터가 비어있다면
        break;                            // 현재 노드에 새로만들어진 노드를
    }

    // 현재 노드의 오른쪽과 왼쪽의 노드들이 모두 값을 갖고 있는 경우
    else
    {
        // 현재 노드의 왼쪽 노드의 왼쪽이 비어있거나 오른쪽 노드가 비어있다면
        if ( pCurrent->pLeft->pLeft == NULL ||
             pCurrent->pLeft->pRight == NULL )
        {
            pCurrent = pCurrent->pLeft;   // 현재 노드를 왼쪽 노드로 옮긴다
        }
        // 현재 노드의 오른쪽 노드의 왼쪽이 비어있거나 오른쪽 노드가 비어있다면
        else if ( pCurrent->pRight->pLeft == NULL ||
                  pCurrent->pRight->pRight == NULL )
        {
            pCurrent = pCurrent->pRight;  // 현재 노드를 오른쪽 노드로 옮긴다
        }

        // 현재 노드의 왼쪽 노드와 오른쪽 노드의 왼쪽, 오른쪽 양방향이 다 차있다면
        else
            pCurrent = pCurrent->pLeft;   //완전트리를 위해 왼쪽 노드로 이동한다
    }
}
if ( pCurrent->pLeft == NULL )            // 현재 노드의 왼쪽이 비어있다면
{
    pCurrent->pLeft = pTemp;              // 새로만들어진 노드를 왼쪽에 연결시킨다
}
else if( pCurrent->pRight == NULL )       // 현재 노드의 오른쪽이 비어있다면
{
    pCurrent->pRight = pTemp;             // 새로만들어진 노드를 오른쪽에 연결시킨다
}
```

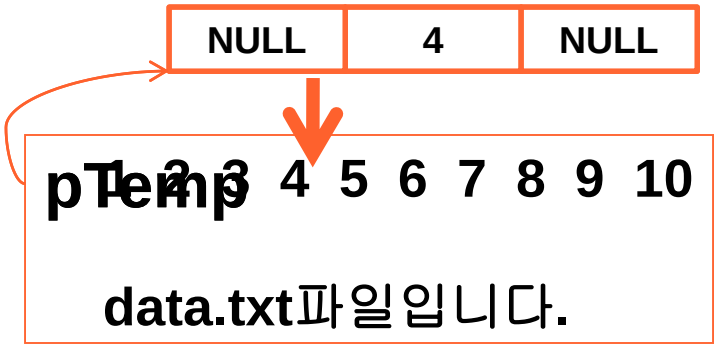


주어야 함으로 while문 통과

링크드 리스트 실습 Solution

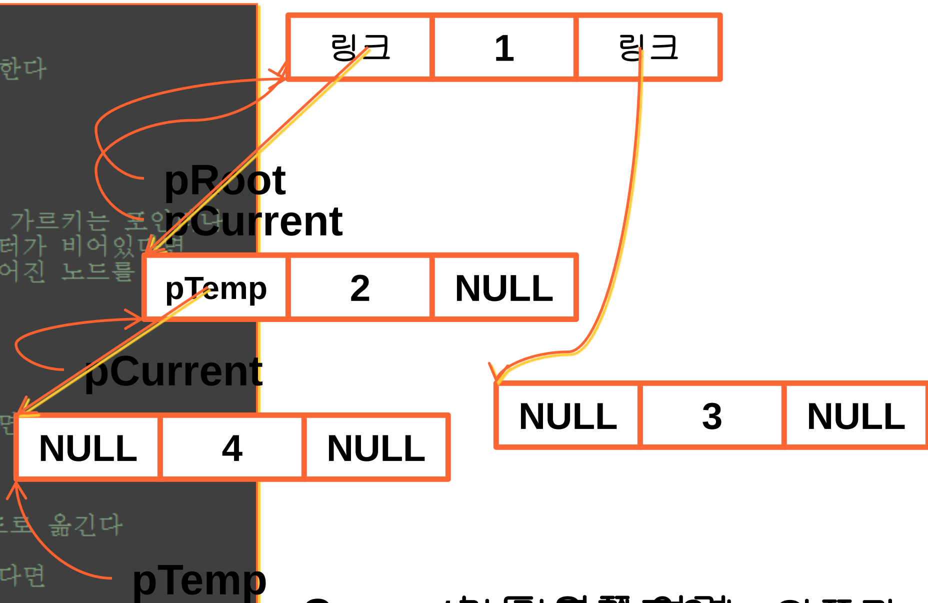
- 파일을 읽어 들이는 코드

```
// 파일의 끝까지 돌면서 노드를 생성하고 트리를 완성시킨다
while ( !feof(pFile) )
{
    int nTemp;                                          // 파일의 숫자를 받아오기위한 변수
                                                        // 구조체 할당을 위한 임시 트리노드 포인터변수
    TreeNode *pCurrent;                                 // 부모를 알기위한 트리노드 포인터 변수

    if ( fscanf( pFile, "%d", &nTemp ) )                // fscanf읽기 성공일때 0외의 값, 읽기 실패일때 0
    {
        // 새로운 트리노드의 생성
        pTemp = (TreeNode*)malloc( sizeof(TreeNode) );  // 트리노드 사이즈만큼 동적할당
        pTemp->data     = nTemp;                        // 새로만들어진 트리노드의 data에 파일의 숫자를 입력
        pTemp->pLeft    = NULL;                         // 새로만들어진 트리노드의 왼쪽 포인터값 초기화
        pTemp->pRight   = NULL;                         // 새로만들어진 트리노드의 오른쪽 포인터값 초기화
    }
```



링크드 리스트 실습 Solution

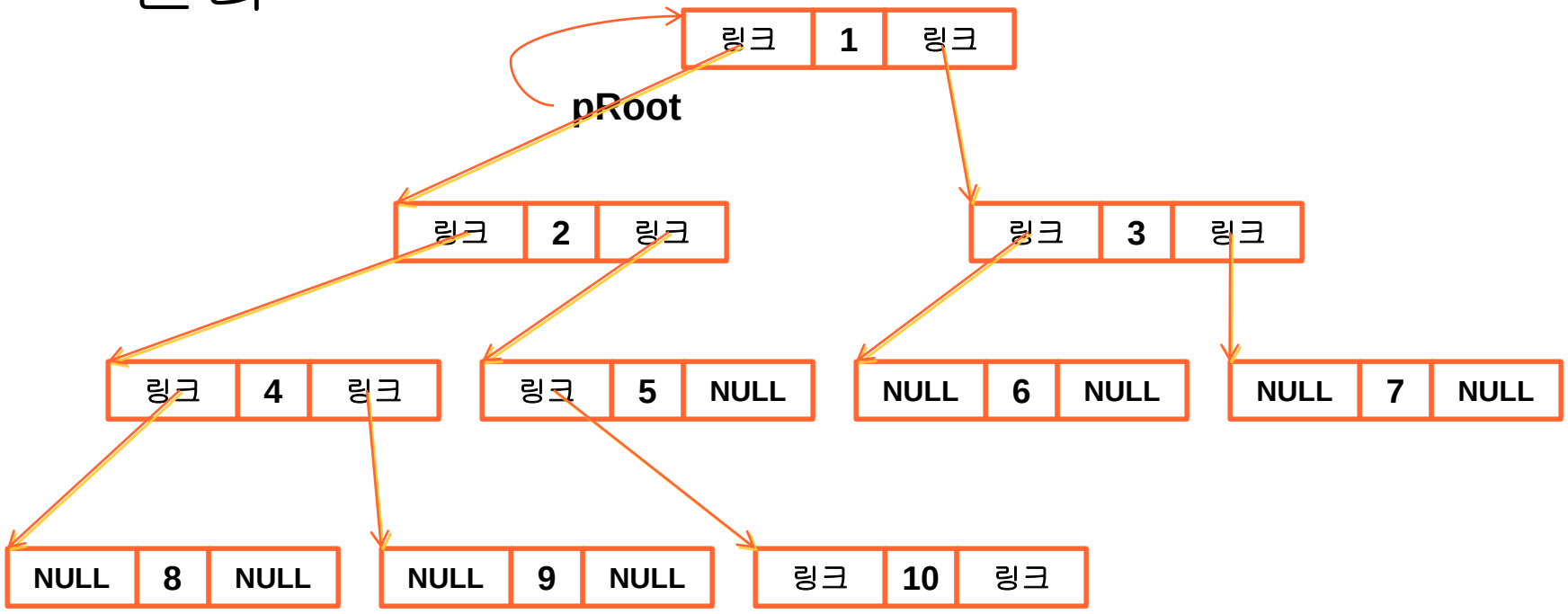
```
// 새로운 노드생성 부분 {.....}

pCurrent = pRoot;                          // 부모는 root부터 시작한다

// 부모가 누가 될것인지 알아내기 위해 순환
while ( pCurrent != NULL )
{
    if ( pCurrent->pLeft == NULL ||
         pCurrent->pRight == NULL )          // 현재 노드의 오른쪽을 가르키는 포인
    {                                        // 왼쪽을 가르키는 포인터가 비어있다면
        break;                               // 현재 노드에 새로만들어진 노드를
    }

    // 현재 노드의 오른쪽과 왼쪽의 노드들이 모두 값을 갖고 있는 경우
    else
    {
        // 현재 노드의 왼쪽 노드의 왼쪽이 비어있거나 오른쪽 노드가 비어있다면
        if ( pCurrent->pLeft->pLeft == NULL ||
             pCurrent->pLeft->pRight == NULL )
        {
            pCurrent = pCurrent->pLeft;      // 현재 노드를 왼쪽 노드로 옮긴다
        }
        // 현재 노드의 오른쪽 노드의 왼쪽이 비어있거나 오른쪽 노드가 비어있다면
        else if ( pCurrent->pRight->pLeft == NULL ||
                  pCurrent->pRight->pRight == NULL )
        {
            pCurrent = pCurrent->pRight;     // 현재 노드를 오른쪽 노드로 옮긴다
        }

        // 현재 노드의 왼쪽 노드와 오른쪽 노드의 왼쪽, 오른쪽 양방향이 다 차있다면
        else
            pCurrent = pCurrent->pLeft;      //완전트리를 위해 왼쪽 노드로 이동한다
    }
}
if ( pCurrent->pLeft == NULL )               // 현재 노드의 왼쪽이 비어있다면
{
    pCurrent->pLeft = pTemp;                 // 새로만들어진 노드를 왼쪽에 연결시킨다
}
else if( pCurrent->pRight == NULL )          // 현재 노드의 오른쪽이 비어있다면
{
    pCurrent->pRight = pTemp;                // 새로만들어진 노드를 오른쪽에 연결시킨다
}
```



노드의 왼쪽을 조사

링크드 리스트 실습 Solution

- 위와 같은 방법으로 파일의 끝까지 읽게되면 아래 그림과 같이 완전 이진트리가 생성되게 된다



링크드 리스트 실습 Solution

- 완전이진트리의 왼쪽 부분 출력하자

```
// 출력부분 : 트리의 왼쪽 부분만 출력한다
pTemp = pRoot;
while ( pTemp != NULL )
{
    printf("%d\t", pTemp->data);          // 현재 노드의 data값을 출력한다
    pTemp = pTemp->pLeft;                 // 현재 노드를 왼쪽 포인터 부분으로 옮긴다
}
```