# Data Structures

## Lab # 10

Data & Knowledge Engineering Laboratory
Kyung Hee University, Korea

Chapter 8 Programming Exercise

# Lab 10

- 1. Exercise 22
- 2. Exercise 26
- 3. Exercise 27
- 4. Exercise 32
- 5. Exercise 35
- 6. Exercise 36

# 실습 준비

- **샘플 코드에 구현된 트리를 사용**
  - ...₩labplus_CRLF₩labplus₩Lab, C++ 3rd₩Chapter8₩Recursive Tree
  - Chapter8중 Recursive 방식을 사용한 트리를 사용

- **실습 문제에 해당하는 함수를 클래스 선언문에(TreeType.h) 추가하고, TreeType.cpp에 해당 함수를 구현**

- **사용할 파일**
  - QueType.h, QueType.cpp, TreeType.h, TreeType.cpp

- **5,6번 문제 : 첨부파일 확인**
  - TreeType.h와 QueType.h 상단에 typedef char ItemType -> typedef int ItemType으로 변경->변경안해도 무관
  - 5번문제 : SortedType.h파일 사용 6번문제 : UnSortedType.h파일 사용

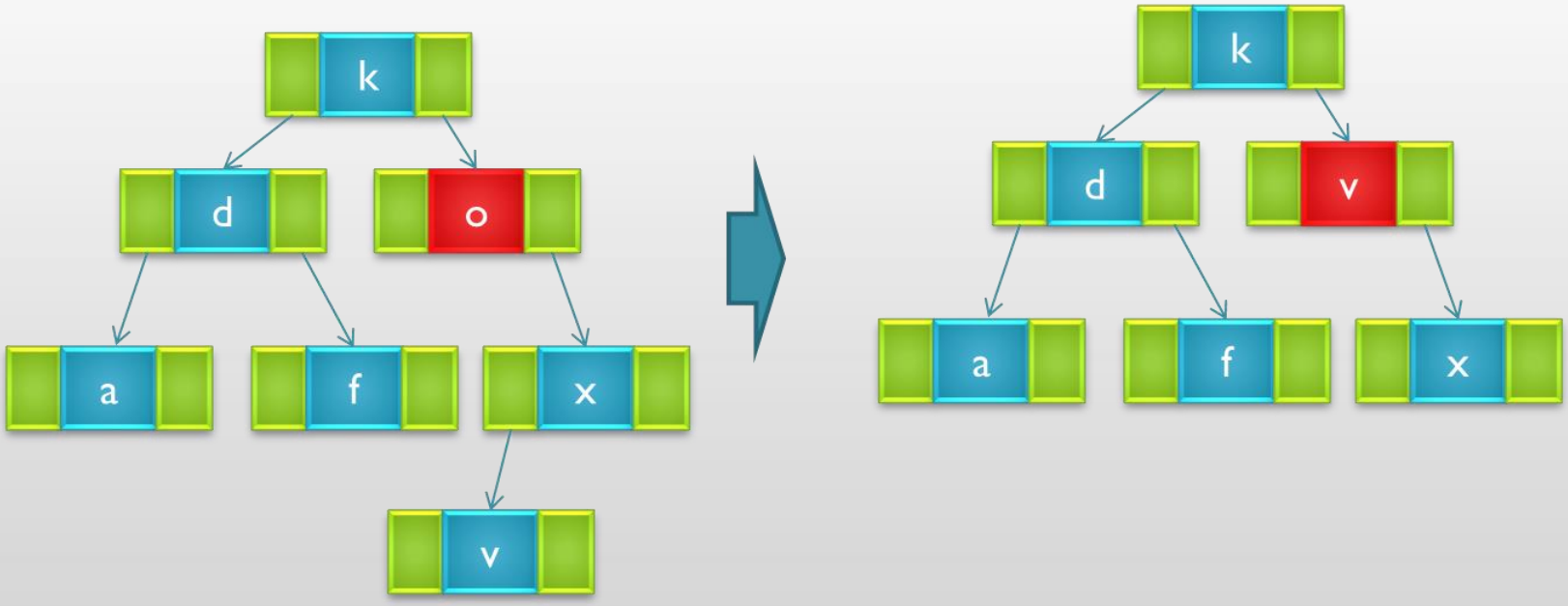
# 1. Exercise 22

## ■ 문 제

❖ 두 자식 노드를 가진 노드를 삭제할 경우에 삭제되는 값의 중간 후임자(전임자가 아닌)를 사용하도록 이 장의 DeleteNode 함수를 수정하라. 또한 이전 연습문제에서 작성한 PtrToSuccessor 함수를 호출하라.

- 중간 후임자 (immediate successor) 삭제하려는 값 다음으로 작은 값

## ■ 예 제




❖ O를 삭제한 경우 O의 위치에 중간 후임자가 위치하게 한다.

# 1-help slide

- 삭제하려는 노드의 right에 위치한 서브 트리에서 가장 작은 값을 찾으시오

```
void DeleteNode(TreeNode*& tree)  //이미 구현된 소스에 수정하세요.
{
  ItemType data;
  TreeNode* tempPtr;

  tempPtr = tree;
  if (tree->left == NULL)
  {
    tree = tree->right;
    delete tempPtr;
  }
  else if (tree->right == NULL)
  {
    tree = tree->left;
    delete tempPtr;
  }
  else
  {
          //이 부분을 전임자가 아닌 중간 후임자의 값이 들어가도록 수정.
          //삭제하려는 노드의 오른쪽에서 PtrToSuccessor()를 사용한다.
          //값을 대치하고 노드 삭제.
  }
}
```

이전 문제에서는 PtrToSuccessor()를 멤버함수로 구현하였는데, DeleteNode() 함수는 멤버함수가 아니므로 멤버함수를 호출할 수 없다. 따라서, PtrToSuccessor()의구현을 비 멤버함수로 변경하여 사용해야 한다.

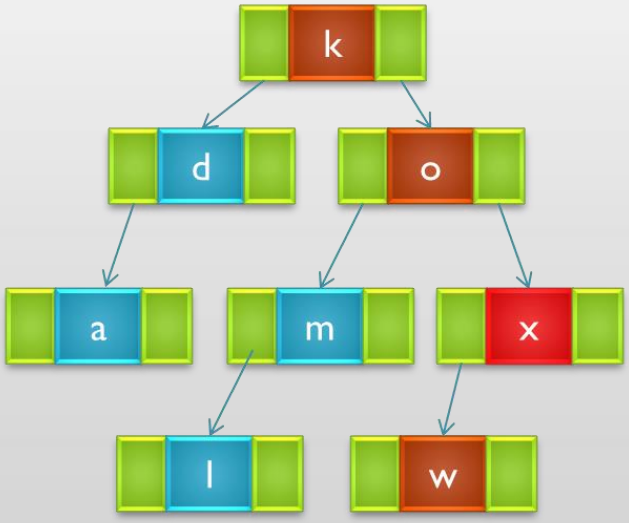
# 2. Exercise 26

## ■ 문 제

- ❖ TreeType멤버 함수로 value값을 기준으로 조상노드에 저장된 모든 값을 출력하는 Ancestors 함수를 작성하라 (단, value값은 출력하지 않음)
- ❖ 함수 프로토타입을 정의하고, 반드시 반복 버전(Iterative)으로 작성하시오.

## ■ 예 제

- ❖ Value = 'x', tree.Ancestors(Value)




k, o 를 출력
(x는 출력하지 않음에 주목)

# 2-help slide

```
//입력 받은 값을 만날 때까지 노드를 따라가면서 내용을 출력해준다.

void TreeType::Ancestors(ItemType value)
{
	bool found=false;
	QueType path;

	TreeNode *currentNode = root;

	while(_____)  // value를 가진 노드를 찾거나 트리 끝에 도달할 때까지
	{
		if(_____)  // value를 가진 노드를 찾으면
			found = true;
		else {
			path.Enqueue(_____); // path에 현재 노드의 값을 삽입
			if(_____)
					_____; // 왼쪽 서브트리로 이동
			else
					_____; // 오른쪽 서브트리로 이동
		}
	}

	if (found)
		// path에 있는 내용을 출력
	else
		// value를 찾을 수 없다고 출력
}
```

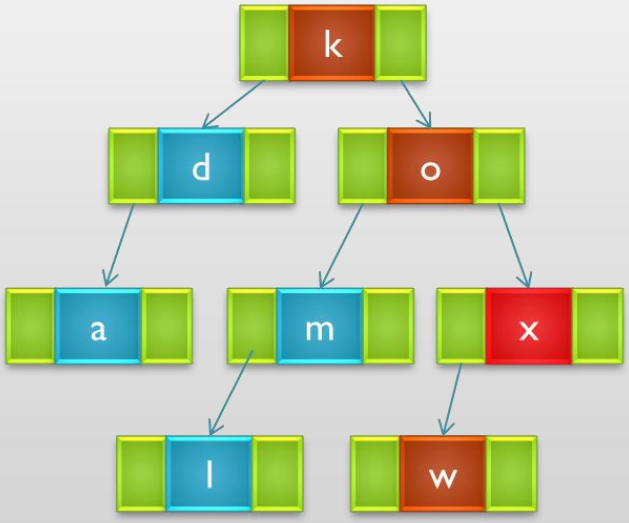
# 3. Exercise 27

## ■ 문 제

❖ 2번 문제에서 작성한 Ancestors함수를 재귀 함수로 작성하시오.

## ■ 예 제

❖ Value = 'x', tree.Ancestors_re(Value)




k, o 를 출력
(x는 출력하지 않음에 주목)

# 3-help slides

```
bool Ancestors_recursive(TreeNode* tree, ItemType item)
{
        bool found = false;
        if(tree == NULL)
                return false;

        if (_____________) // value를 찾은 경우
                return true;

        if (_____________)
                found = Ancestors_recursive(___, item); // 왼쪽 서브트리 호출
        else
                found = Ancestors_recursive(___, item); // 오른쪽 서브트리 호출

        if(found) // 재귀호출 후 value값을 찾았으면
                // 현재 노드의 값을 출력

        return found;
}

void TreeType::Ancestors_re(ItemType value)
{
        Ancestors_recursive(root, value);
}
```

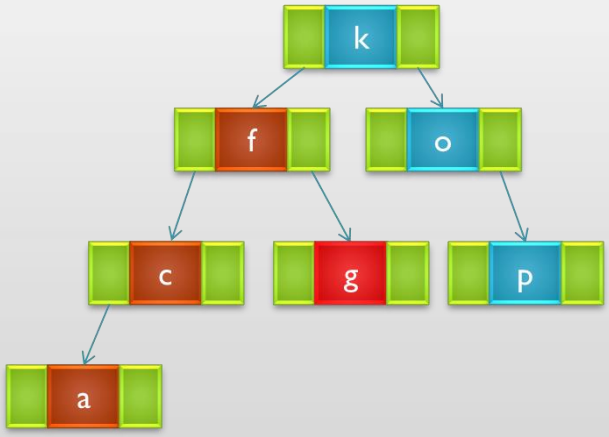
# 4. Exercise 32

## ■ 문 제

❖ 트리내에서 Value 보다 작은 값을 갖는 노드의 수를 반환하는 클라이언트 함수 Smaller를 작성하라.

## ■ 예 제

❖ Value = 'g', Smaller(Value)




'g'를 입력 받을 경우 g보다 작은 a, c, f의 개수인 3이 리턴되야 한다.

# 4-help slide

```
//트리로부터 아이템을 순차적으로 얻으면서 그 값의 개수를 카운트 한다.
//주어진 value보다 큰 아이템을 얻을 경우 즉시 루프를 벗어나도록 한다.

int Smaller(TreeType tree, int value)
{
        ItemType item;
        bool finished = false;
        int count=0;

        tree.ResetTree(IN_ORDER);

        do {
                tree.GetNextItem(_____,_____,_____);

                if (_____)
                        count++;
                else
                        finished = true;
        } while (!finished);

        return count;
}
```

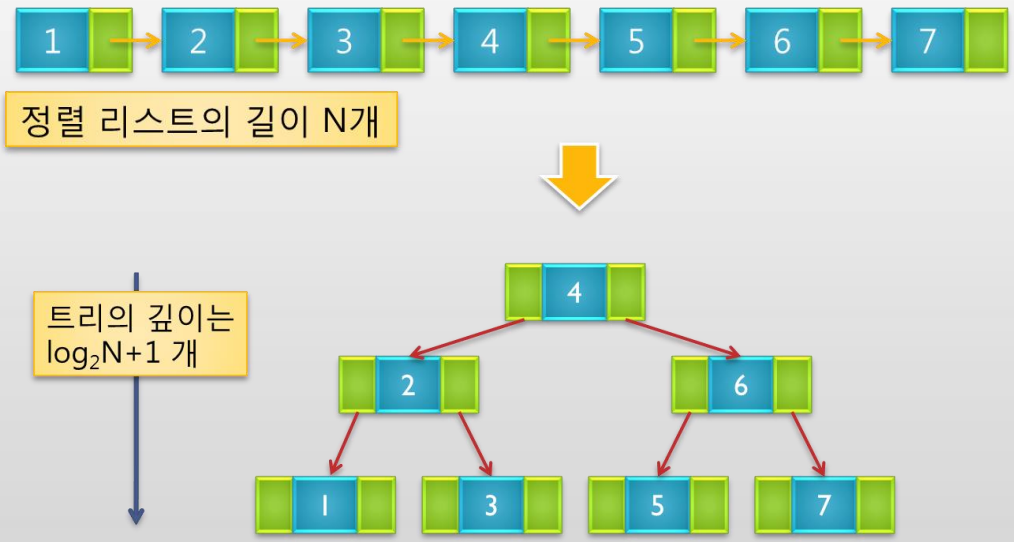
# 5. Exercise 35

## ■ 문 제

❖ 정수형의 정렬 리스트 요소로부터 이진 검색 트리를 생성하는 MakeTree 클라이언트 함수를 작성하라.

- 만약 순차적으로 입력한다면 N레벨의 트리를 만들게 된다. 이점을 유의하여 최대 $\log_2 N + 1$ 레벨을 가지는 트리를 만들어야 한다.

## ■ 예 제



# 5-help slide

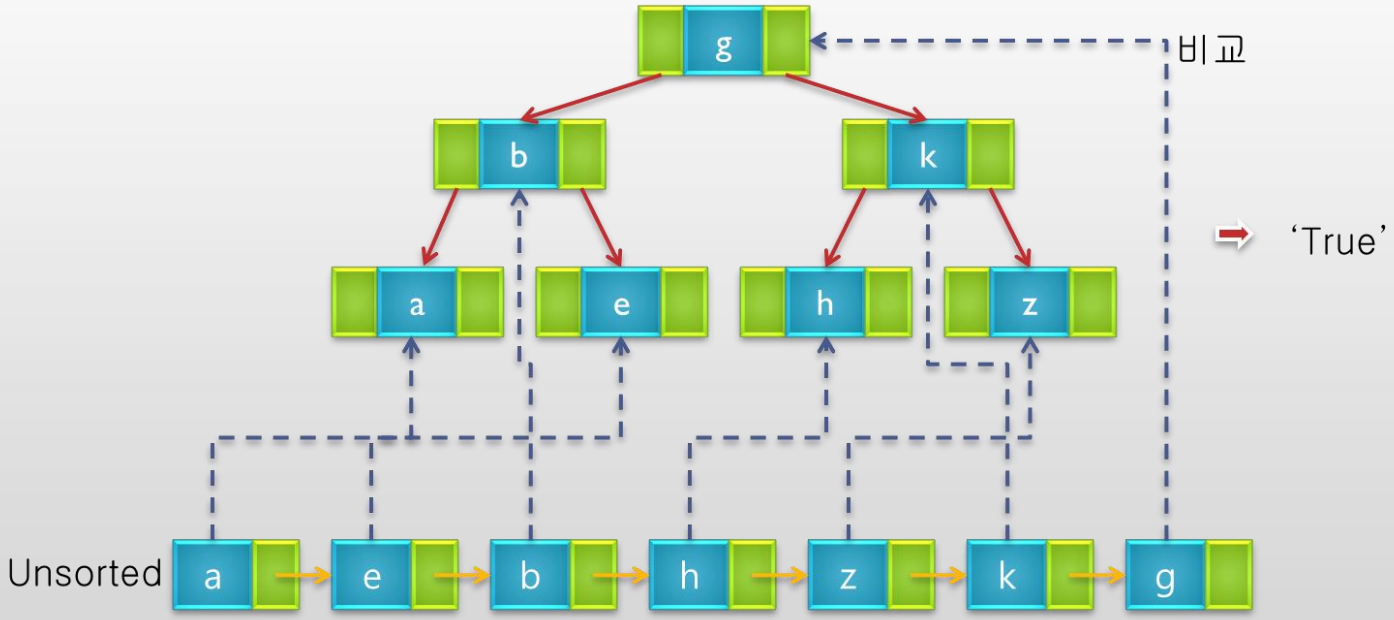
```
//리스트가 가지는 아이템을 배열로 옮긴 뒤, 배열에서 입력 순서를 계산하여 트리에 입력한다.

void AddElement(TreeType& tree,int Array[],int from,int to);
void MakeTree(TreeType &tree, SortedType<int> &list);

void MakeTree(TreeType &tree, SortedType<int> &list)
{

        int length = list.LengthIs();  //리스트 크기를 얻는다
        int * array = new int[length];  //동적 배열 할당
        int item_info;
        int i;

        list.ResetList();

        for(i=0; i < length; i++)
        {
                list.GetNextItem(item_info);
                array[i] = item_info;
        }

        AddElement(tree, array, 0, length-1);

        delete [] array;  // 동적 배열 삭제
}
void AddElement(TreeType& tree,int Array[],int front, int rear)
{
        int mid;                          //중간 값을 기록하는 변수
        if(___________){                  //함수의 종료조건
                mid = _____________;
                tree.InsertItem(Array[mid]);
                AddElement(tree,Array,_____,_____);
                AddElement(tree,Array,_____,_____);
        }
}
```

# 6. Exercise 36

## ■ 문 제

❖ 이진 검색 트리와 비정렬 순차 리스트가 동일한 값들을 가지는지를 결정하는 MatchingItem 클라이언트 함수를 작성하라.

## ■ 예 제




List와 tree의 아이템을 꺼내가면서 비교한다

# 6-help slide

```
bool MatchingItem_Unsorted(TreeType &tree, UnsortedType<ItemType> &list)
{
	int list_length = list.LengthIs();
	int tree_length = tree.LengthIs();

	if(list_length != tree_length) //길이를 먼저 비교, 같은지 검사한다.
	{
		return false;
	}else
	{
		ItemType item;
		bool found;

		__________; //list에 iterator를 사용할 준비를 한다
		for (int i = 0; i < list_length; i++) {
			___________; // list에서 하나의 아이템을 가져온다
			tree.RetrieveItem(item, found); //트리에 해당 아이템이 있는지 검색. O(logN)이 걸림.
			if (!found)) _________________;
		}
		return true;
	}
}
//아래 구현과 비교해 볼 때 수행시간 면에서 무슨 차이가 있을까?
		tree.ResetTree(IN_ORDER); // tree에 iterator를 사용할 준비를 한다
		for (int i = 0; i < list_length; i++) {
			tree.GetNextItem(item, IN_ORDER, found); // tree에서 하나의 아이템을 가져온다
			list.RetrieveItem(item, found); // list에 해당 아이템이 있는 검색. O(N)이 걸림.
			if (!found))		_________________;
		}
```