동국대학교
dongguk university

# 실습: Week 7

Data Structures

# 시험 공지

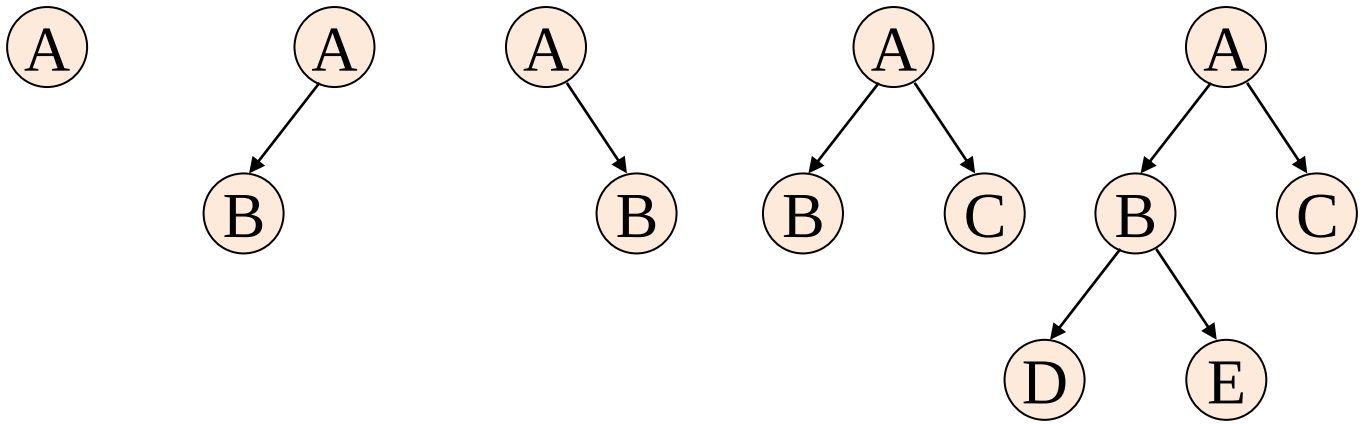
- 날짜 : 4/25 (화)
- 시간 : 13:00 - 15:00
- 장소 : 신공학관 6144 강의실(대면)

# Contents

- Tree
  - Binary tree 구현
  - Binary tree 노드수, 높이, 데이터합 등 계산
  - Binary tree 순회
- 실습
  - 실습 8-1. Binary tree

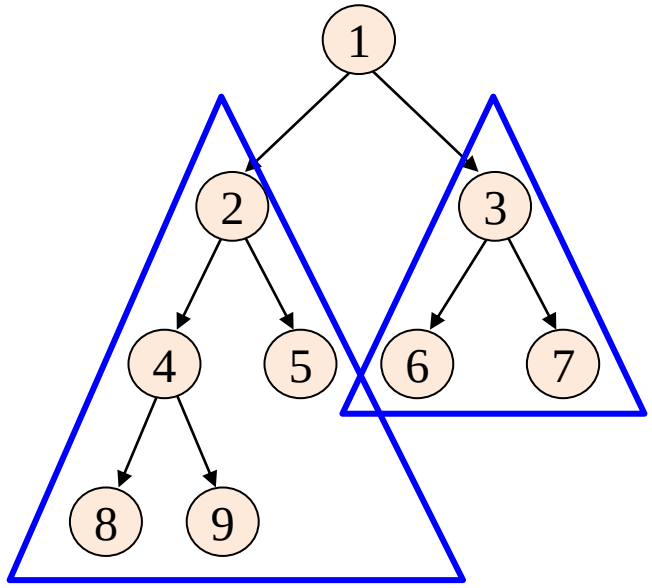
# Binary Trees

- 이진 트리
  - 노드의 유한집합으로서,
    1. 공집합이거나
    2. 루트와 왼쪽 서브트리, 오른쪽 서브트리 두 개의 분리된 이진 트리로 구성됨



# Binary Trees

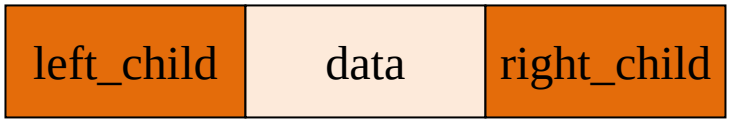
- 이진 트리의 노드 수
  - 왼쪽 서브트리의 노드 수 + 오른쪽 서브트리의 노드 수 + 1
- 이진 트리의 높이
  - Max(왼쪽 서브트리의 높이, 오른쪽 서브트리의 높이) + 1
- 이진 트리의 데이터 합
  - 왼쪽 서브트리의 데이터 합
    + 오른쪽 서브트리의 데이터 합
    + 루트 노드의 데이터



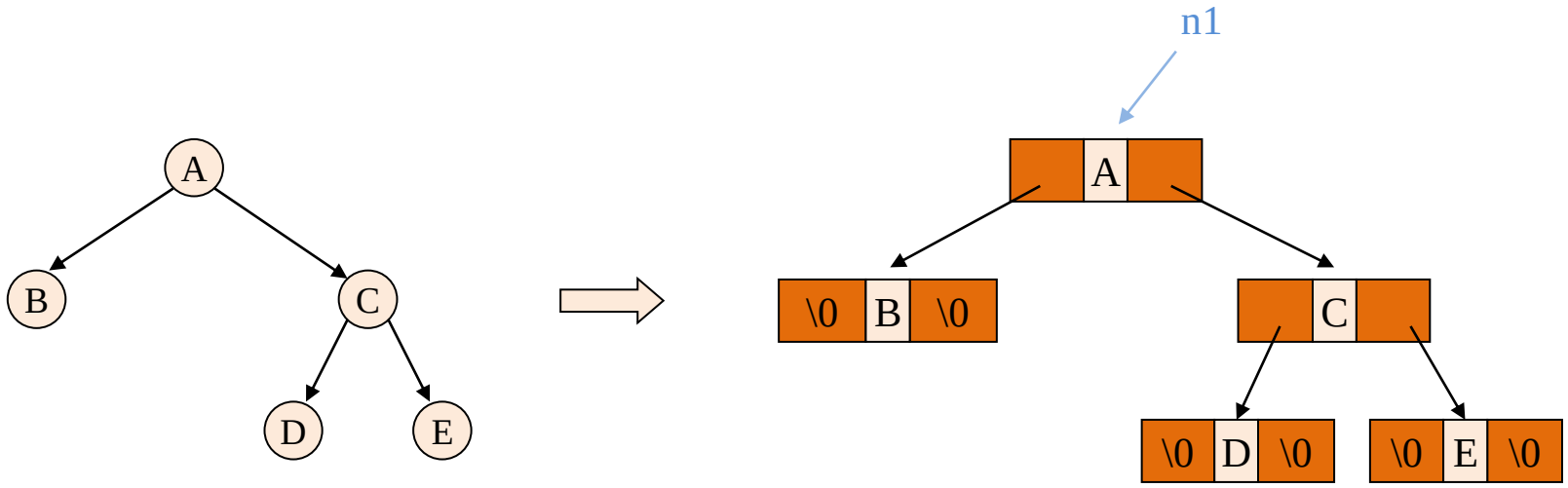
# Linked Representation of BT

- Representation
  - Pointer를 이용하여 parent에 child를 연결

```
typedef struct node *tree_pointer;
typedef struct node{
        int             data;
        tree_pointer    left_child;
        tree_pointer    right_child;
}node;
```



# Linked Representation of BT

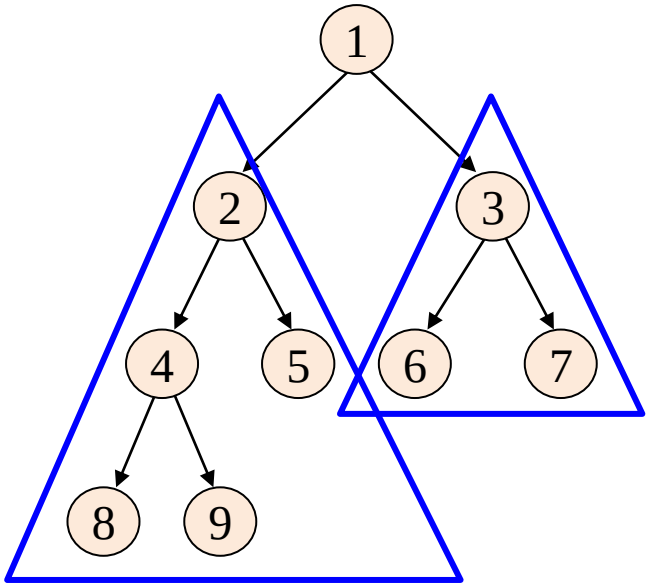



```
tree_pointer n1, n2, n3, n4, n5;

n1 = (tree_pointer)malloc(sizeof(tree_node));
n2 = (tree_pointer)malloc(sizeof(tree_node));
...
n1->data = A; n1->left = n2; n1->right = n3;
n2 ->data = B; n2 ->left = NULL; n2->right = NULL;
...
```

# BT Traversal

- 전위순회(preorder traversal)

```
void preorder(tree_pointer ptr)
{
    if(ptr) {
        printf("%d", ptr->data);
        preorder(ptr->left_child);
        preorder(ptr->right_child);
    }
}
```



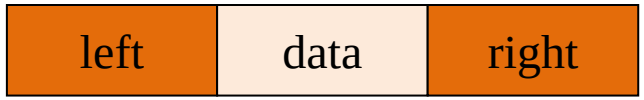

⇨ 1, (2,4,8,9,5), (3,6,7)

# 실습 8-1. binary tree 구현

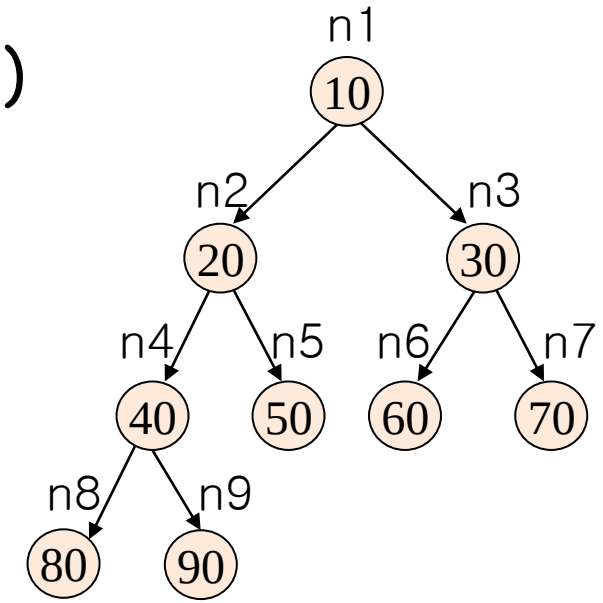
- Binary tree 관련 프로그램
  - 아래와 같은 명령을 수행하는 함수들을 재귀적인 형태로 구현
- 명령어
  - C: Count tree nodes
  - A: Sum tree data
  - H: Height of tree
  - S: Show tree (preorder)
  - F: Free tree
  - Q: Quit

# 자료구조 및 함수

```
typedef struct tree_node *tree_pointer;
typedef struct tree_node {
        Element         data;
        tree_pointer    left;
        tree_pointer    right;
} tree_node ;
```




- tree_pointer build_simple_tree()
  - 다음과 같은 이진 트리를 생성
  - n1~n9의 tree_pointer와 malloc 사용
  - root node 주소값 반환

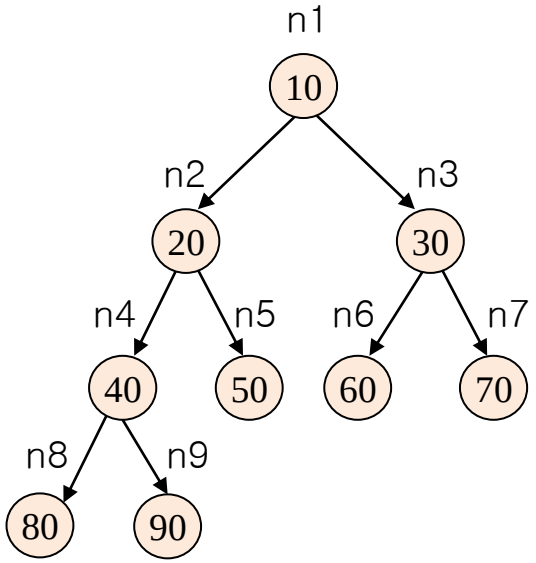


# 자료구조 및 함수

- int bt_count(tree_pointer ptr)
  - 트리의 노드수를 계산
- int bt_sum(tree_pointer ptr)
  - 트리의 데이터 합을 계산
- int bt_height(tree_pointer ptr)
  - 트리의 높이를 계산
  - stdlib.h 에 구현되어 있는 max 함수 활용
- void bt_show_preorder(tree_pointer ptr)
  - 트리의 내용을 preorder로 출력
- void bt_free(tree_pointer ptr)
  - 트리의 모든 노드를 시스템에 반환(free)

# 실습 8-1. 실행 예

```
************* Command ************
C: Count tree, A:  Sum tree data
H: Height of tree, S: Show preorder
F: Free tree, Q: Quit
**********************************

Command> c
 Total number of node = 9

Command> a
 Sum of tree data = 450

Command> h
 Height of tree = 4

Command> s
10      20      40      80      90      50      30      60      70

Command> f
free node with item 80
free node with item 90
free node with item 40
free node with item 50
free node with item 20
free node with item 60
free node with item 70
free node with item 30
free node with item 10
```



# binary_tree.h

```
#include<stdio.h>
#include<stdlib.h>

#define    boolean    int
#define    true       1
#define    false      0

typedef int Element;

// Binary tree node
typedef struct tree_node *tree_pointer;
typedef struct tree_node {
                        Element             data;
                        tree_pointer        left;
                        tree_pointer        right;
} tree_node;

tree_pointer build_simple_tree();   // 앞에 정의된 9개의 노드를 갖는 binary tree를 생성
int bt_count(tree_pointer ptr); // 트리의 노드수를 계산
int bt_sum(tree_pointer ptr); // 트리의 데이터 합을 계산
int bt_height(tree_pointer ptr); // 트리의 높이를 계산
void bt_show_preorder(tree_pointer ptr); // 트리의 내용을 preorder로 출력
void bt_free(tree_pointer ptr); // 트리의 모든 노드를 시스템에반환(free)
```

# binary_tree.c - main()함수

```
#include "binary_tree.h"

void main()
{
        char            c;
        int             n;
        tree_pointer    t;

        t = build_simple_tree();

        printf("************** Command ************\n");
        printf("C: Count tree, A:  Sum tree data    \n");
        printf("H: Height of tree, S: Show preorder \n");
        printf("F: Free tree, Q: Quit               \n");
        printf("************************************\n");

        while (1) {
                printf("\nCommand> ");
                c = _getche();
                c = toupper(c);
```

# binary_tree.c - main()함수

```
switch (c) {
                    case 'C':
                            n = bt_count(t);
                            printf("\n Total number of node = %d \n", n);
                            break;
                    case 'A':
                            n = bt_sum(t);
                            printf("\n Sum of tree data = %d \n", n);
                            break;
                    case 'H':
                            n = bt_height(t);
                            printf("\n Height of tree = %d \n", n);
                            break;
                    case 'S':
                            printf("\n");

                            bt_show_preorder(t);
                            printf("\n");
                            break;
```

# binary_tree.c - main()함수

```
            case 'F':
                        printf("\n");
                        bt_free(t);
            case 'Q':
                        printf("\n");
                        exit(1);
            default: break;
            }
    }
}
```