AppData\Local\Temp\ef5b627a-beea-4664-bce8-815ce32bc2c0_OSAP_003_7_최종보고서(소스코드 포함).zip.2c0\src\find.cc

```
/*
MIT License
This file is part of the INHA_OSAP_003_7 project.
Copyright (c) 2024 tbmyong

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

작성자: 장태환
작성일(파일 생성일): 2024-12-03
작성일(파일 최종 수정일): 2024-12-17
*/

/*<Find 기능 구현>
Node* FindNode(Node* current_node, int key)
: 루트부터 key와의 비교 후 재귀적으로 알맞은 노드포인터값을
  탐색하는 함수. 탐색 실패시 nullptr을 반환.

int CalculateDepthHeightSum(Node* current_node, int key, int ancestor_sum)
: 루트부터 key와의 비교 후 재귀적으로 깊이와 높이의 합을 계산하는 함수
  탐색 실패시 0을 반환.

int Find(int key)
: root와 key값으로 SumDepthHeight 호출
*/

#include "../base/avl.h"

Node* AVL::FindNode(Node* current_node, int key) {
  // 노드가 없는 경우
  if (current_node == nullptr) return nullptr;

  // 노드의 키값이 찾는 key와 같은 경우
  if (current_node->get_key() == key) return current_node;
  // 노드의 키값이 찾는 key보다 큰 경우
  else if (current_node->get_key() > key)
```

```
52      return FindNode(current_node->get_left(), key);
53    // 노드의 키값이 찾는 key보다 작은 경우
54    else
55      return FindNode(current_node->get_right(), key);
56  }
57
58  int AVL::CalculateDepthHeightSum(Node* current_node, int key) {
59    int depth = 0;
60
61    // 노드가 nullptr이 아닌 경우 반복
62    while (current_node != nullptr) {
63      // 노드의 키값이 찾는 key와 같은 경우
64      if (key == current_node->get_key()) {
65        return depth + current_node->get_height();
66      }
67      // 노드의 키값이 찾는 key보다 큰 경우
68      if (key < current_node->get_key()) {
69        current_node = current_node->get_left();
70      }
71      // 노드의 키값이 찾는 key보다 작은 경우
72      else {
73        current_node = current_node->get_right();
74      }
75      depth++;
76    }
77    return 0;
78  }
79
80  int AVL::Find(int key) { return CalculateDepthHeightSum(root_, key); }
```