OOP Project 보고서

컴퓨터정보공학부 2021202043 이은서

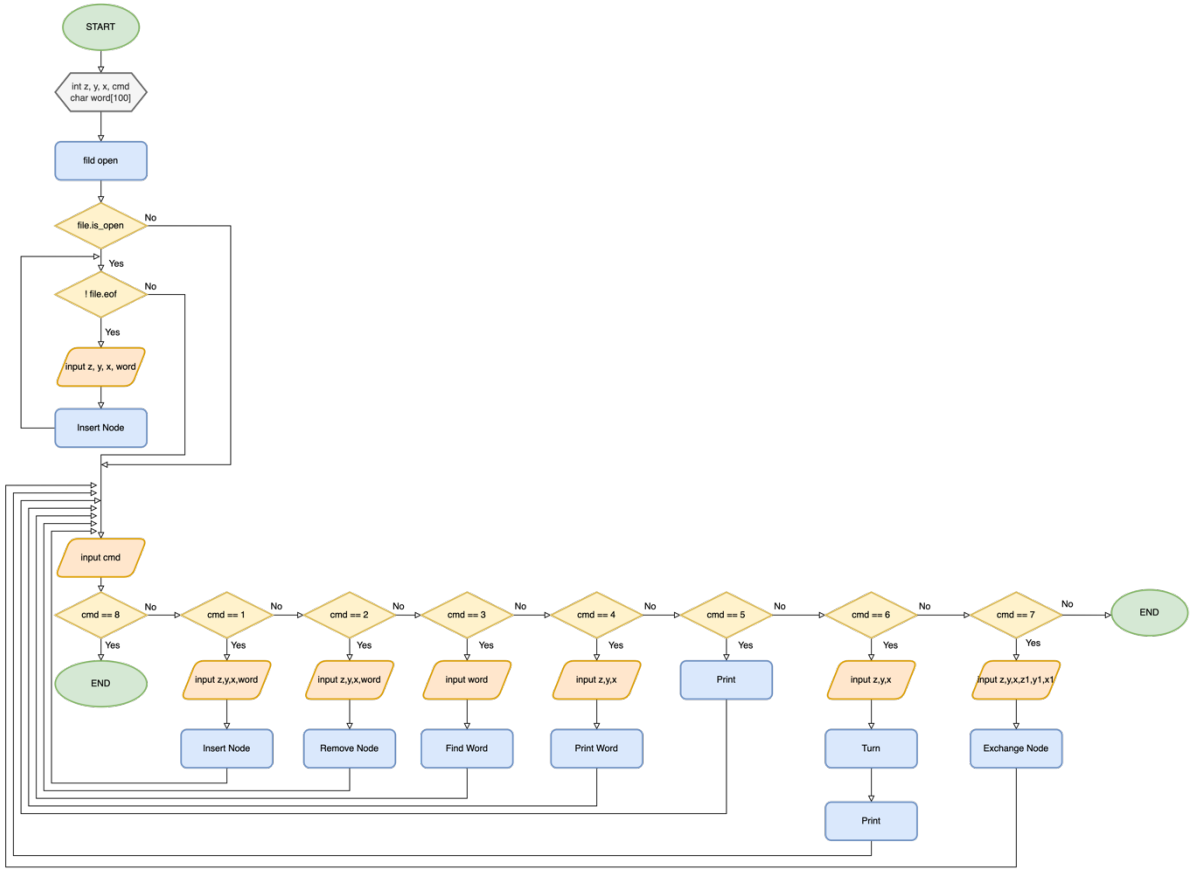
1. Introduction

각 면이 회전할 수 있고 Block끼리 교환할 수 있는 3x3x3 크기의 3D 큐브를 구성하여 큐브 내부에 단어를 저장한다. 즉, 총 27개의 블록을 가지고 있고, 각각은 3d 링크드리스트로 연결되어 있다. 그리고 각각의 블록에 단어를 저장하고 있다고 하였는데, 이 단어는 bst 구조로 사전식 배열을 기준으로 저장되어 있다. 그리고 이 bst를 이용하여 단어를 저장, 삭제, 탐색, 출력이 가능하다. 클래스를 이용하여 링크드리스트를 구현하고, 각각의 노드는 bst라는 링크드리스트와도 연결한다. 하나의 노드는 prev, next, forward, backward, up, down 총 6개의 방향으로 다른 노드들과 연결되어있고, 전면 좌상단을 영점으로 (z, y, x)의 좌표값을 가지고 있다. 즉, 각 노드별로 z, y, x의 좌표 값을 저장해야 한다. 그렇게 3d 3x3x3 큐브를 구성한 다음, 단어를 저장하기 위해 "WordBook.txt"를 불러온다. 이때, 파일입출력을 사용하여 한줄씩 불러와서 각 줄에 적혀있는 z, y, x좌표와 word를 구분하고 이를 노드에 저장한다. 좌표를 기준으로 노드를 찾은 다음, 그 노드의 bst에서 단어를 삽입하면 된다. Delete는 좌표를 찾고, 그 단어가 존재하면 단어를 삭제한다. 이때 사전식 정렬을 유지하기 위해, 각 단어 노드의 왼쪽, 오른쪽 자식 노드를 불러와서 경우의 수를 나눈다. 첫 번째 경우는 왼쪽, 오른쪽 모두 자식 노드가 없고, 자신이 마지막 노드일 경우이고, 두 번째 경우는 왼쪽, 혹은 오른쪽 둘 중 한 쪽에만 자식 노드가 있는 경우이다. 이때는 삭제할 단어의 부모 노드를 삭제할 단어의 하나만 있는 자식노드와 연결시킨 다음, 자기 자신을 지우면 된다. 세 번째 경우는 왼쪽, 오른쪽 모두 자식 노드가 있는 경우이다. 이때는 삭제할 단어부터 시작해서 제일 작은 단어, 즉 제일 왼쪽에 있는 단어를 찾은 다음, 그 단어를 자기 자신의 위치에 복사한 다음, 바꾼 단어를 제거하면 된다. Find는 0,0,0 좌표부터 2,2,2 좌표까지 모든 노드에 들리면서 해당 단어가 있는지 찾고, 있으면 그 좌표를 출력하게끔 만들면 된다. Print함수도 해당 좌표를 찾은 다음, 그 노드에 접근하여 모든 단어를 출력하게끔 만들면 된다. Print_all에서는 해당 블록에서 저장하고 있는 단어의 개수를 출력해야하는데, 개수를 저장하는 변수를 따로 만들어서, 처음에 단어를 삽입할 때, 개수를 저장하는 변수만 1씩 증가시키면 된다. 마찬가지로 삭제할 때는 1씩 감소하게 한 다음, 모든 블록에 접근하면서 개수를 출력하게끔 만들면 된다. Turn은 한 면을 회전해야 하므로, 무슨 축을 기준으로 돌리는지 전면, 측면, 후면으로 3가지 경우를 먼저 나누고, 거기서 시계방향과 반시계 방향 2가지 경우를 나눈 다음, 각각의 block의 연결을 다시 해주면 된다. Exchange의 경우에는 temp에다가 하나 저장한 다음, 두 개의

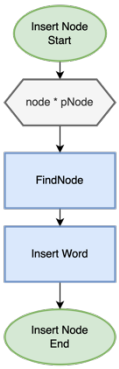
블록의 위치를 교환하면 된다. 이때, 내부 bst를 교환하는 것이 아닌, block 자체를 교환해야 한다.

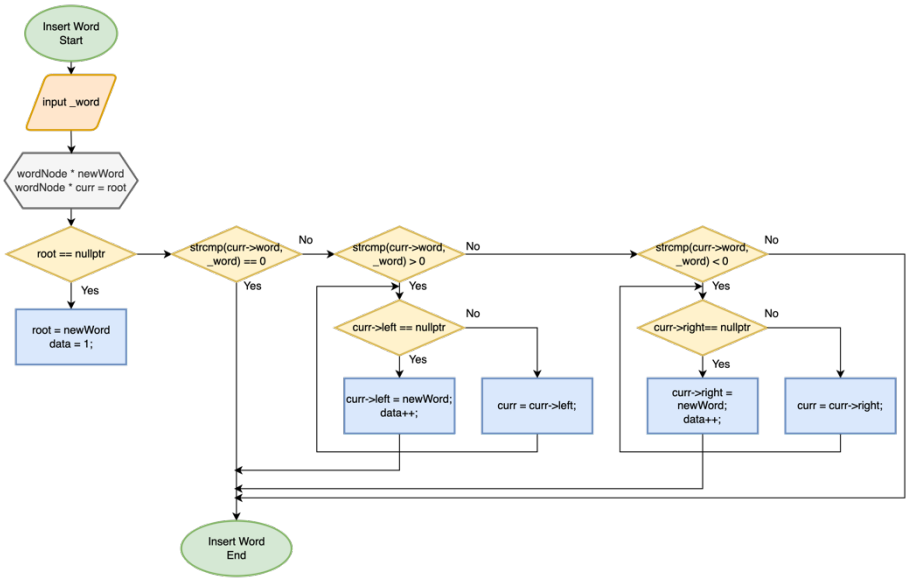
Flowchart

- **전체 Flow Chart**

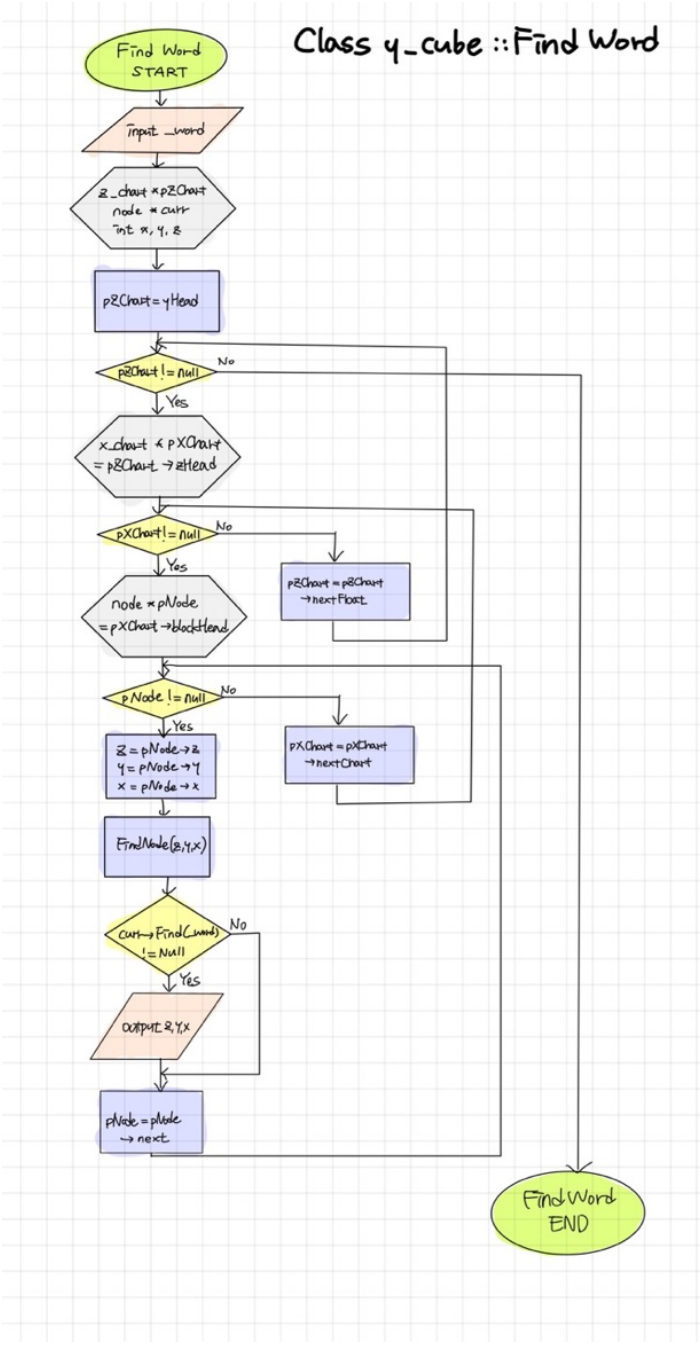


- Insert Node, Insert Word


Insert Node Start
node * pNode
FindNode
Insert Word
Insert Node End

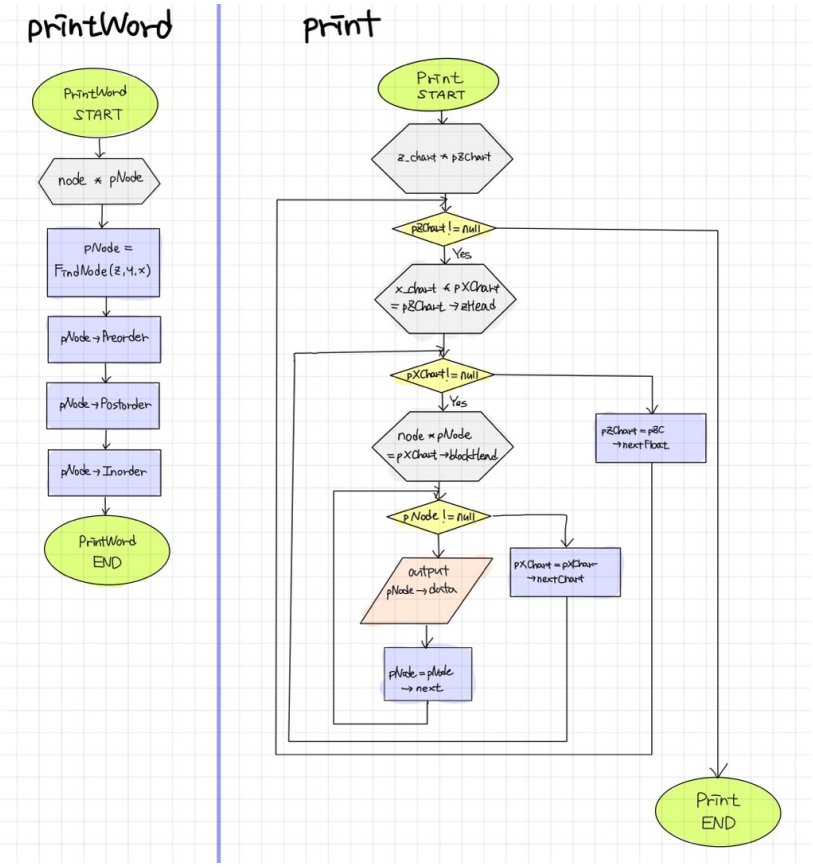

Insert Word Start
input _word
wordNode * newWord
wordNode * curr = root
root == nullptr
Yes
root = newWord
data = 1;
strcmp(curr->word, _word) == 0
No
Yes
strcmp(curr->word, _word) > 0
No
Yes
curr->left == nullptr
No
Yes
curr->left = newWord;
data++;
curr = curr->left;
strcmp(curr->word, _word) < 0
No
Yes
curr->right== nullptr
No
Yes
curr->right = newWord;
data++;
curr = curr->right;
Insert Word End

- Find Word

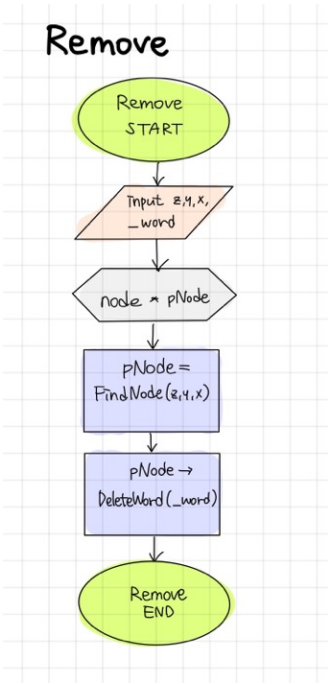


-

- PrintWord, Print



- Remove

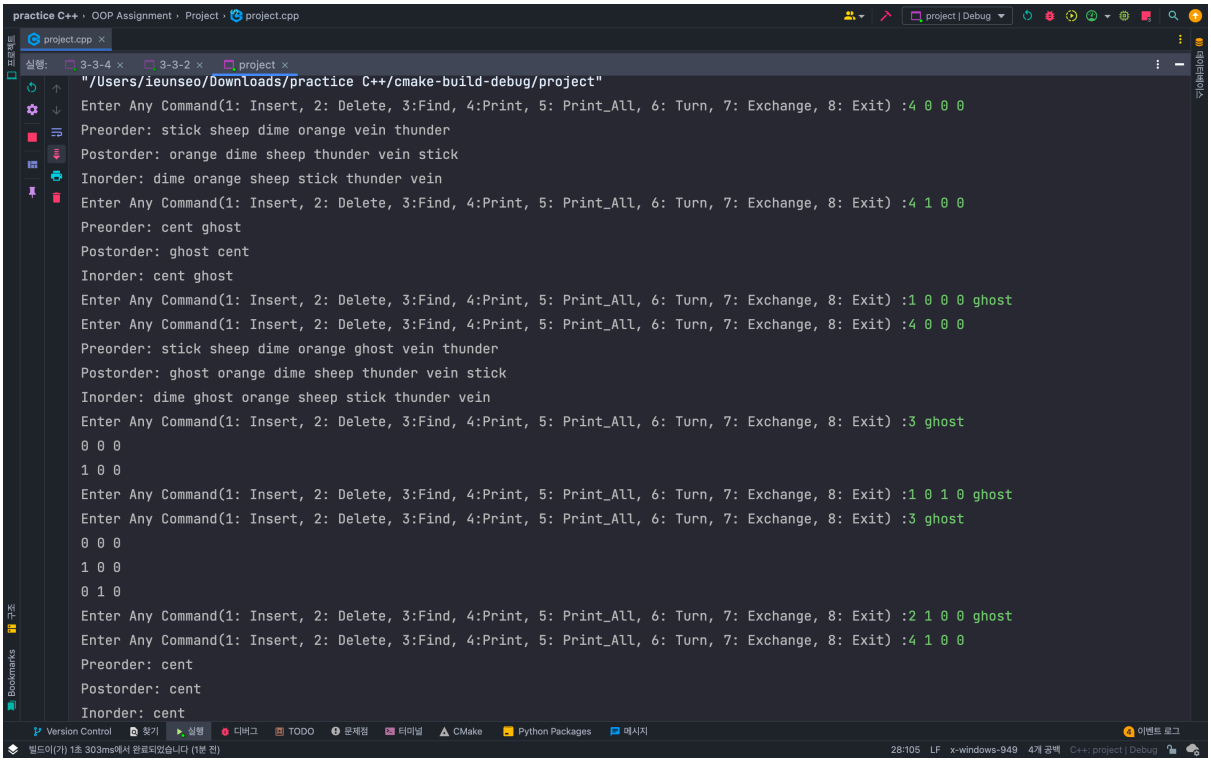


2. Algorithm

cube 탐색은 0,0,0 좌표에서부터 2,2,2좌표까지 각각의 x축, z축, y축의 헤드에서부터

next로 이동해가면서 매게변수로 입력받은 좌표의 값과 현재 노드의 좌표 값이 같을 때, 그 노드의 주소를 return한다. 그리고 return한 값을 기준으로 print, 혹은 insert에 이용하였다. 3D Linked List을 구성하기 위해 가장 먼저 한 일은, 블록을 생성하는 것이다. 그리고, 그 블록을 3번 생성하고, 각각의 블록의 next, prev를 연결하는 x_chart class를 생성하였다. 그리고, 그 x_chart를 3번 생성하는 z_chart class를 생성하고, x_chart에 있는 블록에 각각 접근하여 up, down을 연결하였다. 그 다음에는 z_chart를 3번 생성하는 y_cube class를 생성하고, z_chart에 있는 x_chart를 하나씩 접근하면서 각각의 block에 접근하여 forward, backward를 연결시킨다. 그러면 각각의 노드마다 next, prev, backward, forward, up, down이 연결되어 있다. BST는 word를 저장하는 노드와, 그 word node들을 통합하여 관리하는 wordlist class를 각각 구성하였고, wordlist class에서 단어들의 삽입, 찾기, 삭제, 출력을 모두 구현하였다. Insert의 경우에는 curr = root로 두고, root가 null일 때는 아무 단어도 저장되어 있지 않은 것이므로, root에 저장할 단어를 연결하고, 그 block의 단어 개수 변수를 1로 설정한다. Root가 비어져 있지 않은 경우에는 root와 insert할 단어를 비교하고, 만약 이미 있는 단어이면 무시, 그렇지 않으면 입력할 단어가 더 작으면 왼쪽으로, 입력할 단어가 더 크면 오른쪽으로 이동하면서 더이상의 자식 노드가 없을 때까지 이동한다.그리고, 그 비어있는 노드에 저장하고, block의 단어 개수 변수를 +1 한다. Delete는 맨 처음 curr=root로 두고, root부터 더이상의 노드가 없을 때까지 비교하고, 만약 삭제할 단어가 현재 단어보다 작으면 왼쪽으로, 크면 오른쪽으로 이동하면서 같은 단어가 나올때까지 이동한다. 그리고 같은 단어가 나올 때까지 이동하면, 3가지의 경우를 나눈다. 왼쪽, 오른쪽 자식 노드 모두 비어있는 경우, 한쪽만 비어있는 경우, 두 쪽 모두 자식노드가 있는 경우이다. 모두 비어있는 경우에는 그냥 freet시켜버리고, 한 쪽만 비어있는 경우에는 하나 남아있는 자식 노드를 삭제할 노드의 부모 노드와 연결한 다음 자기 자신을 free하여 지운다. 두 자식 노드 모두 있는 경우에는 삭제할 노드를 기준으로 가장 작은 노드, 즉 가장 왼쪽에 있는 노드를 자신의 자리에 복사한 다음 subtree의 가장 왼쪽 노드를 제거한다.

3. Result & Verification




```
Enter Any Command(1: Insert, 2: Delete, 3:Find, 4:Print, 5: Print_All, 6: Turn, 7: Exchange, 8: Exit) :1 0 0 0 ghost
Enter Any Command(1: Insert, 2: Delete, 3:Find, 4:Print, 5: Print_All, 6: Turn, 7: Exchange, 8: Exit) :1 0 1 0 ghost
Enter Any Command(1: Insert, 2: Delete, 3:Find, 4:Print, 5: Print_All, 6: Turn, 7: Exchange, 8: Exit) :2 1 0 0 ghost
Enter Any Command(1: Insert, 2: Delete, 3:Find, 4:Print, 5: Print_All, 6: Turn, 7: Exchange, 8: Exit) :5
7       4       4
1       2       5
5       4       4

4       4       4
2       1       1
3       2       6

5       2       4
2       4       3
3       10      5
```

4. Conclusion

3d linked list를 구성할 때 노드를 생성하고 연결하는 함수를 따로 만들지 않고, 생성자에서 모두 행해지게 만들었다. 그 결과 코드를 조금 더 간단하게 만들 수 있었다. 그리고 insert, delete, print, find에서 공통적으로 사용하는 것이 무엇일지 고민해보았고, 해당 좌표를 갖는 노드를 가져오는 함수를 만들면 더 효율적이 될거라고 생각하여 좌표를 입력받으면, 그 좌표의 노드를 return하는 findNode함수를 만들어서 insert, delete, find, print, print_all 등 모든 명령에서 사용하게끔 하였다. Exchange함수는 temp를 만든 다음 2개의 좌표를 findNode함수를 사용하여 저장하고, 교환하였지만, 포인터 값만 교체되고 연결은

제대로 이루어지지 않았다. Exchange 함수를 구현할 수 없다 보니 turn은 도저히 생각할 수 없었다. Temp를 이용하여 포인터 값을 교환하는 방법이 틀렸거나 아예 다른 방법으로 접근해야 하는 것 같은데 잘 모르겠다.