1. 코드 분석 및 주석

#include <stdio.h>

typedef struct TreeNode {//트리 노드를 위한 구조체 선언

int data;

struct TreeNode \*left, \*right, \*parent;//트리 구조를 위한 포인터 변수들

} TreeNode;

// G

// C F

// A B D E

TreeNode n1 = { 'A', NULL, NULL, NULL };//트리 구조체에 데이터 넣기

TreeNode n2 = { 'B', NULL, NULL, NULL };

TreeNode n3 = { 'C', &n1, &n2, NULL };

TreeNode n4 = { 'D', NULL, NULL, NULL };

TreeNode n5 = { 'E', NULL, NULL, NULL };

TreeNode n6 = { 'F', &n4, &n5, NULL };

TreeNode n7 = { 'G', &n3, &n6, NULL };

TreeNode \*exp = &n7;

TreeNode \*tree\_successor(TreeNode \*p) {

TreeNode \*h;

h = p->right;//p의 right 주소를 담는다.

if (h != NULL) {//h's right is not NULL

while (h->left != NULL)//return the leftmost node of right subtree

h = h->left;

return h;

}

TreeNode \*y = p->parent;//parent 주소를 담는다.

while (y != NULL && p == y->right) {//parent가 null이 아니고 p가 parent의 right child 일 경우

p = y;

y = y->parent;//계속 위의 parent를 찾는다.

}

return y;

}

void main()

{

TreeNode \*q = exp;

n1.parent = &n3;

n2.parent = &n3;

n3.parent = &n7;

n4.parent = &n6;

n5.parent = &n6;

n6.parent = &n7;

n7.parent = NULL;

while (q->left) q = q->left; // Go to the leftmost node. 가장 왼쪽에 있는 것 부터 시작해서 successor

do

{

printf("%c\n", q->data); // Output data

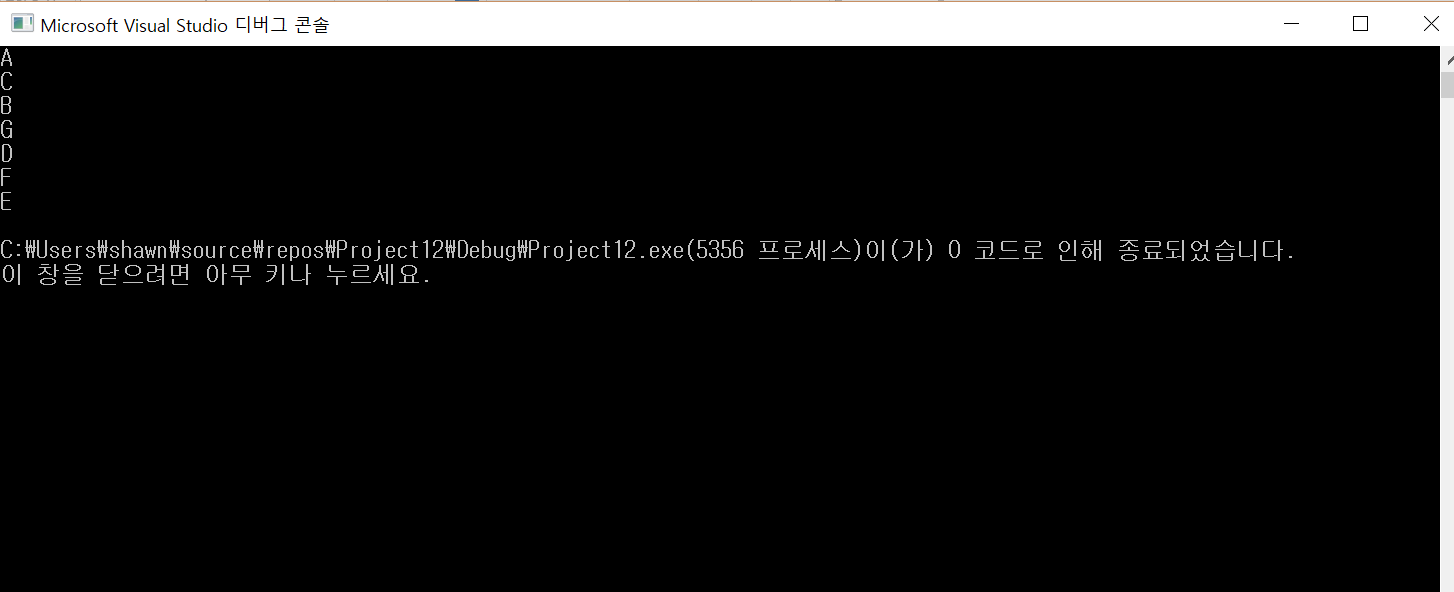
// Call the successor

q = tree\_successor(q);

} while (q); // If not null

}

2. 결과 콘솔 창



3. 주요 코드 분석

1) treenode 구조체에서 중요한 점은 right child와 left child를 선언하는 것. 하지만 successor를 찾기 위해서는 parent node 또한 추가하였다.

TreeNode n1 = { 'A', NULL, NULL, NULL }

2) 만약 right subtree가 존재한다면 successor는 그 right subtree 중 가장 왼쪽에 있는 노드이다. 그래서 leftnode가 존재하지 않을 때 까지 while 문을 돌리면서 맨 왼쪽에 있는 노드에 접근하도록 하였다.

h = p->right;//p의 right 주소를 담는다.

if (h != NULL) {//h's right is not NULL

while (h->left != NULL)//return the leftmost node of right subtree

h = h->left;

return h;

}

3) 만약 right subtree가 존재하지 않는 경우라면 parent를 찾아간다. 하지만 바로 위의 parent는 아니고 더 이상 parent 가 존재하지 않을 때 까지 찾아나선 다음에 맨 마지막에 있는 parent가 succesor이다.

TreeNode \*y = p->parent;//parent 주소를 담는다.

while (y != NULL && p == y->right) {//parent가 null이 아니고 p가 parent의 right child 일 경우

p = y;

y = y->parent;//계속 위의 parent를 찾는다.

}

return y;