1. 코드 분석 및 주석

#include <stdio.h>

typedef struct TreeNode

{

int data;

struct TreeNode \*left, \*right, \*parent;

} TreeNode

;

//G

// C F

// A B D E

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

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\_predecessor(TreeNode \*p) {

TreeNode \*h;

h = p->left;//p의 left 주소를 담는다.

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

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

h = h->right;

return h;

}

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

while (y != NULL && p == y->left) {//parent가 null이 아니고 p가 parent의 left 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->right) q = q->right;

// Go to the rightmost node

do {

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

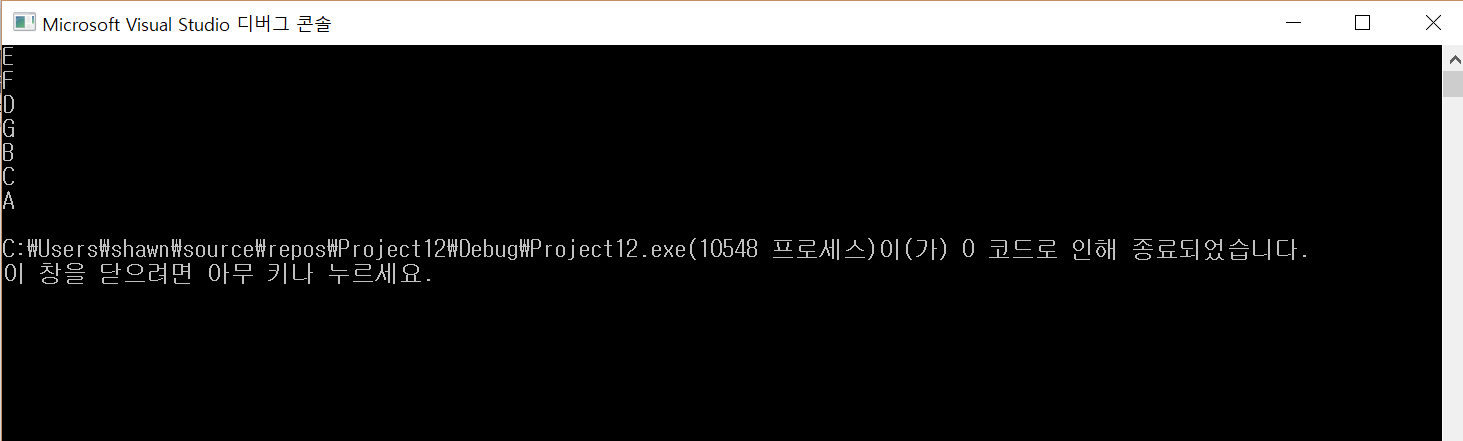
// Call the predecessor

q = tree\_predecessor(q);

} while (q); // If not null

}

2. 결과 콘솔 창



3. 주요 코드 분석

1) successor를 찾는 코드와 매우 비슷하지만 다른 점은 바로 left를 모두 right으로 바꿔야 한다는 것이다. 만약 노드의 leftsubtree가 존재한다면 predecessor는 rightmost node이다.

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

TreeNode \*h;

h = p->left;//p의 left 주소를 담는다.

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

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

h = h->right;

return h;

}

2) 만약 node의 left subtree가 없는 경우에는 parent를 찾아봐야 한다. 하지만 만약 node가 parent의 right child가 아닌 left child 일 경우는 부모의 부모를 찾아나서야 한다.

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

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

p = y;

y = y->parent;//계속 위의 parent를 찾는다.

}

return y;

}

3) succesor가 아닌 predecessor를 찾는 경우이기 때문에 출력은 가장 마지막 노드, 즉 rightmost node 부터 해야 한다.

while (q->right) q = q->right;