# MakeTree(num\_of\_input);

讀取時，以陣列的方式逐一讀取，若非-1，則繼續向左subtree加入新的節點。若為第一個-1，則將下一個非-1的值放進右subtree；若是連續兩個緊鄰的-1，則pop到上一層的右subtree。

# preorder(num\_of\_input);

程式碼: preorder — parent-> leftchild -> rightchild

void preorder(int num){

Node\* cur=BT;

top=-1;

push(cur);

***(以輸入的數字數量以及堆疊做為迭代的條件***

***之後，以current node所鏈結的節點狀況作為條件，)***

for(int i=0; i<num && stack[top]!=NULL;i++){

printf("%d ",cur->key);

if(cur->lchild==NULL){

cur=pop();

if(cur->rchild==NULL&&cur->lchild==NULL){

if(top==-1)break;//終止條件

cur=pop();

cur=cur->rchild;

push(cur);

}else if(cur->rchild!=NULL){

cur=cur->rchild;

push(cur);

}

}else

if(cur->lchild!=NULL){

cur=cur->lchild;

push(cur);

}

}

};

***以current node所鏈結的節點狀況作為條件***

1. 當current node的左subtree 為空，進一步探討右subtree是否為空:

1.1 若左subtree與右subtree皆為空，則pop()到上一層(或更上一層)，並移動到該層的右subtree

1.2 若左subtree為空，但右subtree非空，則前往右subtree，並將這一步記錄到堆疊

2. 當current node的左subtree 非空，則繼續往左並push到堆疊。

終止條件: 當堆疊為空且不能再pop

# postorder(num\_of\_input);

程式碼: postorder — leftchild -> rightchild -> parent

do{

while (cur)

{

if (cur->rchild)

push(cur->rchild);

push( cur);

cur = cur->lchild;

}

cur = pop();

if (cur->rchild && stack[top] == cur->rchild)

{

pop();

push(cur);

cur = cur->rchild;

}

else

{

printf("%d ", cur->key);

cur = NULL;

}

} while (top!=-1); //作為迭代的條件

1. While(cur): 以右子結點、父結點、左子樹右子結點與左子樹父結點的順序將位置紀錄進堆疊直到遇到盡頭(cur=NULL)
2. 接著pop，如果該節點沒有子節點則print 並，若有，只可能是右，所以移動到右節點，並回到while 找出所有該節點之下的節點，並放入堆疊。