자료구조 과제 9

* –

20151523

김동현

- 실행결과

개체, 시계, 플레이어이(가) 표시된 사진

자동 생성된 설명

- 코드 및 알고리즘

#include<stdio.h>

#include<stdlib.h>

#define true 1

#define false 0

typedef struct threadedTree \*threadedPointer;

typedef struct threadedTree {

short int leftThread;

threadedPointer leftChild;

char data;

threadedPointer rightChild;

short int rightThread;

}threadedTree;

: 이진 트리 구조체를 정의해준다. Lchild와 rchild로 접근하는 포인터를 설정해준다. 그리고 thread를 설정하여 false인 경우는 자식 노드가 존재하는 경우, true인 경우는 후속자에 접근한다는 것을 나타내준다.

void construct\_tree(threadedPointer tree);

threadedPointer insucc(threadedPointer tree);

void tinorder(threadedPointer tree);

void insertRight(threadedPointer s, threadedPointer r);

void insert(threadedPointer s, char data);

int main()

{

threadedPointer head;

head = (threadedPointer)malloc(sizeof(threadedTree));

head->leftThread = head->rightThread = false;

head->leftChild = head->rightChild = head;

construct\_tree(head);

insert(head->leftChild->rightChild, 'E');

insert(head->leftChild->leftChild->rightChild, 'F');

insert(head->leftChild->leftChild, 'G');

tinorder(head);

getchar();

return 0;

}

: head의 할당을 해준다. 그리고 양 스레드는 false로 선언해주고, 그 자식을 자기 자신으로 해준다. 그리고 construct\_tree를 호출해준다. 그 함수를 통해서 A, B, C, D노드의 배치를 해준다. 그리고 insert함수를 통해 E, F, G 노드를 배치해준다. 그리고 tinorder를 호출하여 출력해준다.

텍스트, 덮여있는, 하얀색, 그룹이(가) 표시된 사진

자동 생성된 설명

<완성된 트리의 모습>

그러므로 inorder의 순서는 B G D F A C E 가 된다.

void construct\_tree(threadedPointer tree)

{

threadedPointer temp, temp\_root;

temp\_root = tree;

/\* insert node 'A' to the left of the head \*/

temp = (threadedPointer)malloc(sizeof(threadedTree));

temp->leftThread = temp->rightThread = true;

temp->leftChild = temp->rightChild = temp;

temp->data = 'A';

temp\_root->leftChild = temp;

temp\_root->leftThread = false;

: head노드의 왼쪽 자식에 A를 data로 갖는 노드를 추가해준다. 그리고 자식을 가리키게 되었으므로 head 노드의 왼쪽 스레드를 false라고 해준다.

temp\_root = tree->leftChild;

/\* insert node 'B' to the left of the temp\_head \*/

temp = (threadedPointer)malloc(sizeof(threadedTree));

temp->rightThread = temp->leftThread = true;

temp->leftChild = tree;

temp->rightChild = temp;

temp->data = 'B';

temp\_root->leftChild = temp;

temp\_root->leftThread = false;

: A노드의 왼쪽 자식에 B를 data로 갖는 노드를 추가해준다. 그리고 자식을 가리키게 되었으므로 A 노드의 왼쪽 스레드를 false라고 해준다.

/\* insert node 'C' to the right of the temp\_head \*/

temp = (threadedPointer)malloc(sizeof(threadedTree));

temp->leftThread = temp->rightThread = true;

temp->leftChild = temp\_root;

temp->rightChild = tree;

temp->data = 'C';

temp\_root->rightChild = temp;

temp\_root->rightThread = false;

: A노드의 오른쪽 자식에 C를 data로 갖는 노드를 추가해준다. 그리고 자식을 가리키게 되었으므로 C 노드의 오른쪽 스레드를 false라고 해준다.

temp\_root = temp\_root->leftChild;

/\* insert node 'D' to the right of the temp\_head \*/

temp = (threadedPointer)malloc(sizeof(threadedTree));

temp->leftThread = temp->rightThread = true;

temp->leftChild = temp\_root;

temp->rightChild = tree->leftChild;

temp->data = 'D';

temp\_root->rightChild = temp;

temp\_root->rightThread = false;

}

: : : B노드의 오른쪽 자식에 D를 data로 갖는 노드를 추가해준다. 그리고 자식을 가리키게 되었으므로 B 노드의 오른쪽 스레드를 false라고 해준다.

텍스트, 하얀색이(가) 표시된 사진

자동 생성된 설명

<construc tree의 결과>

threadedPointer insucc(threadedPointer tree)

{

threadedPointer temp;

temp = tree->rightChild;

if (!tree->rightThread)

while (!temp->leftThread)

temp = temp->leftChild;

return temp;

}

: 중위 후속자를 찾는 함수이다. 만약 오른쪽 스레드가 true인 경우는 그 rightchild가 가리키는 곳이 중위 후속자이므로 바로 반환해주고, 만일 false인 경우 왼쪽 서브 트리에 진입해서 가장 아래에 존재하는 왼쪽 자식을 반환해준다.

void tinorder(threadedPointer tree)

{

threadedPointer temp = tree;

for (; ;) {

temp = insucc(temp);

if (temp == tree) break;

printf("%3c", temp->data);

}

}

: 이진 트리의 노드를 inorder의 순서로 출력해주는 함수이다.

void insert(threadedPointer s, char data)

{

threadedPointer temp;

temp = (threadedPointer)malloc(sizeof(threadedTree));

temp->data = data;

insertRight(s, temp);

}

: 해당 data를 인자로 받아서 트리에 노드를 삽입해주는 함수이다. insertRight 함수를 호출하여 수행한다.

void insertRight(threadedPointer s, threadedPointer r)

{

threadedPointer temp;

r->rightChild = s->rightChild;

r->rightThread = s->rightThread;

r->leftChild = s;

r->leftThread = true;

s->rightChild = r;

s->rightThread = false;

if (!r->rightThread) {

temp = insucc(r);

temp->leftChild = r;

}

}

: r을 s의 오른쪽 자식으로 삽입하는 함수이다. s의 오른쪽 자식의 정보를 먼저 r에 저장해주고, r의 왼쪽 자식에 s를 넣어줌으로서 후속자로 가리키게끔 해준다. 그리고 후속자가 들어갔으므로 스레드는 true로 저장해준다. 그다음 s의 오른쪽 자식에 r을 지정해준다. 그렇게 되면 s의 rightchild포인터는 r을 가리키므로 스레드는 false로 해주고, 만일 r의 오른쪽 스레드가 false라면 원래 s의 아래에 더 있었다는 뜻이므로 r을 삽입 후 V라고 했을 때 중위후속자를 찾아서 그 왼쪽 자식으로 r노드를 후속자로 가리키게 해준다.