**자료구조 OpenLab 5   
Threaded Binary Trees**

**20161637 장호영**

#include<stdio.h>

#include<stdlib.h>

typedef struct threadedTree\* threadedPointer;

typedef struct threadedTree {

short int leftThread;

threadedPointer leftChild;

char data;

threadedPointer rightChild;

short int rightThread;

}threadedTree;

//justift bool type

typedef int bool;

#define true 1

#define false 0

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()

{

/\* initialize a head node \*/

// type your code

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

root->leftChild = NULL;

root->rightChild = root;

root->leftThread = false;

root->rightThread = false;

/\* construct a given tree \*/

construct\_tree(root);

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

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

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

tinorder(root);

//getchar();

printf("\n");

return 0;

}

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 = true;

temp->rightThread = true;

temp->leftChild = temp;

temp->rightChild = temp;

temp->data = 'A';

temp\_root->leftChild = temp;

temp\_root->leftThread = false;

temp\_root = tree->leftChild;

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

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

temp->rightThread = true;

temp->leftThread = true;

temp->leftChild = tree;

temp->rightChild = temp;

temp->data = 'B';

temp\_root->leftChild = temp;

temp\_root->leftThread = false;

/\* 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;

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;

}

threadedPointer insucc(threadedPointer tree)

{

/\* find the inorder successor of tree

in a threaded binary tree \*/

threadedPointer temp;

temp = tree->rightChild;

if (!tree->rightThread)

while (!temp->leftThread)

temp = temp->leftChild;

return temp;

}

void tinorder(threadedPointer tree)

{

//by using insucc function, just repeat printing the data of node

threadedPointer temp = tree;

for (; ; ) {

temp = insucc(temp);

if (temp == tree) break;

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

}

}

void insert(threadedPointer s, char data)

{

//allocate new threadedPointer with given data and call insertRight function to insert the node into tree

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

temp->rightChild = NULL;

temp->rightThread = false;

temp->leftChild = NULL;

temp->leftThread = false;

temp->data = data;

insertRight(s, temp);

}

void insertRight(threadedPointer s, threadedPointer r)

{

/\* insert child as the right child of parent in a threaded binary tree \*/

threadedPointer temp;

r->rightChild = s->rightChild;

r->rightThread = s->rightThread;

r->leftChild = s;

r->leftThread = true;

s->rightChild = r;

s->rightThread = false;

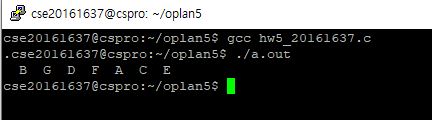
if (!r->rightThread) {

temp = insucc(r);

temp->leftChild = r;

}

}



실행 화면입니다.

이 프로그램은 굉장히 간단합니다. 기존 이진 트리에서 threaded tree로 바꾸어 inorder traversal에 최적화된 자료구조를 만들어 보는 것이 목적입니다.

void construct\_tree(threadedPointer tree)

처음의 트리를 구성해주는 함수입니다. Root A B C D data를 넣어 트리를 구조화합니다.

threadedPointer insucc(threadedPointer tree)

다음으로 올 successor를 찾습니다. 먼저 오른쪽으로 가서 threaded tree라면 반환, 아니라면 왼쪽 움직이며 threaded tree라면 반환합니다.

void tinorder(threadedPointer tree)

inorder 형태로 print하는 함수입니다. 정말 간단하게 insucc함수를 호출하여 print를 찍으면 현재 노드가 root가 되는 순간 멈추면 inorder 이 완성됩니다.

void insert(threadedPointer s, char data)

insertRight를 위해 삽입 위치s와 주어진 data를 토대로 새로운 노드를 할당하고 insertRight 함수를 불러 해당 위치에 넣어줄 수 있도록 합니다.

void insertRight(threadedPointer s, threadedPointer r)

insertRight는 s의 오른쪽 자식에 r을 넣는 함수입니다.

