**자료구조 과제 6**

(쓰레드 이진트리 생성 및 후위순회)

logo.gif

|  |  |
| --- | --- |
| 과목명 | 자료구조 |
| 담당교수 | 김계영 |
| 학과 | 소프트웨어학부 |
| 학년 | 3 |
| 학번 | 20152994 |
| 성명 | 이진영 |
| 제출일 | 2017.6.19 |



**과제6 – 쓰레드 이진트리 생성 및 후위순회**

20152994 이진영

**1. 원시 코드**

|  |
| --- |
| #include <stdio.h>  #include <malloc.h>  #include <string.h>  #define TRUE 1  #define FALSE 0  #define MAX\_NODES 100  char\* treeInfo[] = { "L.A","R.B",  "LL.C","LR.D","RL.E","RR.F",  "LLL.G","LLR.H","LRL.I",'\0' };  int post\_seq = 1;  int first\_in = TRUE;  typedef struct threadedTree \*threadedPointer;  typedef struct threadedTree {  short int leftThread;  threadedPointer leftChild;  char data;  threadedPointer rightChild;  short int rightThread;  };  threadedPointer node;  threadedPointer root;  threadedPointer node0;  threadedPointer child[MAX\_NODES];  threadedPointer postsucc(threadedPointer tree); //후위순회 후속자 리턴  void postorder(threadedPointer tree); //후위순회 시행  void insertLeft(threadedPointer parent, threadedPointer child);  void insertRight(threadedPointer parent, threadedPointer child);  threadedPointer insucc(threadedPointer tree); // 중위순회 후속자 리턴  threadedPointer inPredec(threadedPointer tree); // 중위순회 선행자 리턴  void main() {  root = (threadedPointer)malloc(sizeof(threadedPointer));  node0 = (threadedPointer)malloc(sizeof(threadedPointer));  root->data = '-';  root->rightChild = root;  root->rightThread = FALSE;  root->leftChild = node0;  root->leftThread = FALSE;  node0->data = 'Z';  node0->leftChild = root;  node0->leftThread = TRUE;  node0->rightChild = root;  node0->rightThread = TRUE;  int i = 0;  int num = 0;  while (treeInfo[i]) { // 트리 정보에 알맞게 트리 생성  node = node0;  for (int j = 0; j < strlen(treeInfo[i]); j++) {  if (treeInfo[i][j] == 'L') {  if (node->leftThread == TRUE) {  child[num] = (threadedPointer)malloc(sizeof(threadedTree));  insertLeft(node, child[num]);  num++;  if (treeInfo[i][j + 1] == '.') {  node->leftChild->data = treeInfo[i][j + 2];  j = strlen(treeInfo[i]) - 1;  }  }  node = node->leftChild;  }  if (treeInfo[i][j] == 'R') {  if (node->rightThread == TRUE) {  child[num] = (threadedPointer)malloc(sizeof(threadedTree));  insertRight(node, child[num]);  num++;  if (treeInfo[i][j + 1] == '.') {  node->rightChild->data = treeInfo[i][j + 2];  j = strlen(treeInfo[i]) - 1;  }  }  node = node->rightChild;  }  }  i++;  }  printf("\n※후위순회 결과\n");  postorder(root);  printf("\n\n");  }  threadedPointer postsucc(threadedPointer tree) {//후위순회 후속자 구하는 함수  threadedPointer temp; //3단계로 나눠 단계별 행동 정의하고 함수 호출시마다 1~3단계 반복  if (post\_seq == 1) {  temp = tree;  if (first\_in != TRUE) {  while (!temp->rightThread)  temp = temp->rightChild;  }  temp = temp->rightChild->rightChild;  if (first\_in != TRUE) {  if (temp == root) {  return temp;  }  }  first\_in = FALSE;  while (!temp->leftThread)  temp = temp->leftChild;  post\_seq++;  }  else if (post\_seq == 2) {  temp = tree->rightChild;  if (temp->rightThread == FALSE)  temp = tree->rightChild->rightChild;  post\_seq++;  }  else if (post\_seq == 3) {  temp = tree;  if (temp->leftThread == TRUE) {  temp = tree->leftChild;  }  else if (temp->leftThread == FALSE) {  while (!temp->leftThread)  temp = temp->leftChild;  temp = temp->leftChild;  }  post\_seq = 1;  }  return temp;  }  void postorder(threadedPointer root) {  threadedPointer temp = root;  for (;;) {  temp = postsucc(temp);  if (temp == root) {  printf("%c ", root->leftChild->data);  break;  }  else {  printf("%c ", temp->data);  }  }  }  void insertRight(threadedPointer parent, threadedPointer child)  {  threadedPointer temp;  child->rightChild = parent->rightChild;  child->rightThread = parent->rightThread;  child->leftChild = parent;  child->leftThread = TRUE;  parent->rightChild = child;  parent->rightThread = FALSE;  if (!child->rightThread) {  temp = insucc(child);  temp->leftChild = child;  }  }  void insertLeft(threadedPointer parent, threadedPointer child)  {  threadedPointer temp;  child->leftChild = parent->leftChild;  child->leftThread = parent->leftThread;  child->rightChild = parent;  child->rightThread = TRUE;  parent->leftChild = child;  parent->leftThread = FALSE;  if (!child->leftThread) {  temp = inPredec(child);  temp->rightChild = child;  }  }  threadedPointer insucc(threadedPointer tree)  {  threadedPointer temp;  temp = tree->rightChild;  if (!tree->rightThread)  while (!temp->leftThread)  temp = temp->leftChild;  return temp;  }  threadedPointer inPredec(threadedPointer tree)  {  threadedPointer temp;  temp = tree->leftChild;  if (!tree->leftThread)  while (!temp->rightThread)  temp = temp->rightChild;  return temp;  } |

**2. 실행 결과**

**① 트리 그림**

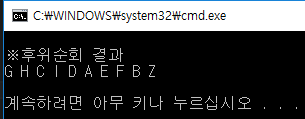
**② 입력 자료**

char\* treeInfo[] = { "L.A", "R.B",

"LL.C", "LR.D", "RL.E", "RR.F",

"LLL.G", "LLR.H", "LRL.I", '\0'};

**③ 실행 결과**

****