

자료구조응용

13. 트리: 이진트리 생성 및 순회(2) (15점)

2020.4.29.(수)

1. 후위표현식(postfix expression)을 입력받아 Figure 5.16과 같은 이진트리를 구성한 후, 이진트리 순회를 통해 중위표현식(infix expression), 전위표현식(prefix expression), 후위표현식(postfix expression)을 출력하는 프로그램을 작성하라. (6점)

(1) 실행순서

- ① 후위표현식으로 부터 산술식의 이진트리를 생성한다. ※ createBinTree();

데이터 입력형식
- 입력파일(input.txt) : AB/C*D*E+
- 피연산자(Operands) : 알파벳 한 글자
- 연산자(Operators) : +, -, *, /, %

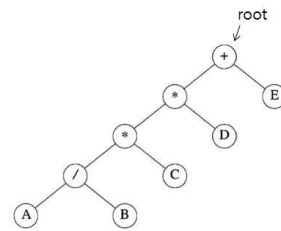


Figure 5.16: Binary tree with arithmetic expression

※ 자료구조응용 09. 1번 프로그램을 수정하여 구현할 수 있음

- eval()을 수정하여 후위표현식을 이진트리로 만드는 createBinTree() 함수를 정의
- getToken() 과 precedence 형은 그대로 사용, stack은 일부 수정
- 구체적인 알고리즘은 아래 구현세부사항에 있음

※ 산술식의 이진트리에서 연산자 노드의 왼쪽 서브트리는 그 연산자의 왼쪽 피연산자로 사용되고 오른쪽 서브트리는 오른쪽 피연산자로 사용됨

- ② 이진트리 중위순회를 통해 중위표현식(infix expression)을 출력한다. ※ inorder(root);
③ 이진트리 전위순회를 통해 전위표현식(prefix expression)을 출력한다. ※ preorder(root);
④ 이진트리 후위순회를 통해 후위표현식(postfix expression)을 출력한다. ※ postorder(root);

(2) 구현 세부사항

```
typedef struct node *treePointer;
typedef struct node {
    char data; // 문자출력을 위해 char 형으로 지정
    treePointer leftChild, rightChild;
}node;
treePointer root;
treePointer stack[MAX_STACK_SIZE];
int top = -1;
char expr[81]; // postfix expression
```

<p>postfix expression으로부터 연결리스트를 사용한 이진트리를 만드는 알고리즘(createBinTree) ※ Program 3.13 eval 함수를 아래 알고리즘으로 수정</p>
<p>왼쪽에서 오른쪽으로 수식을 스캐닝하면서 다음을 수행함</p> <ol style="list-style-type: none"> 1. 토큰이 operand 라면 <ol style="list-style-type: none"> ① 노드를 생성한 후 //※ 노드의 두 링크는 NULL ② stack에 push 2. 토큰이 operator 이면 <ol style="list-style-type: none"> ① 노드를 생성한 후 ② 오른쪽 자식으로 stack에서 pop한 노드를 연결하고 ③ 왼쪽 자식으로 stack에서 또 pop한 노드를 연결한 후 ④ 그 operator 노드를 stack에 push 3. stack에 마지막으로 남은 노드가 root이다.

※ 아래 세 함수에서 printf("%c", ptr->data); 사용

```
void inorder(treePointer ptr)
{
    /* inorder tree traversal */
    if (ptr) {
        inorder(ptr->leftChild);
        printf("%d", ptr->data);
        inorder(ptr->rightChild);
    }
}
```

Program 5.1: Inorder traversal of a binary tree

```
void preorder(treePointer ptr)
{
    /* preorder tree traversal */
    if (ptr) {
        printf("%d", ptr->data);
        preorder(ptr->leftChild);
        preorder(ptr->rightChild);
    }
}
```

Program 5.2: Preorder traversal of a binary tree

```
void postorder(treePointer ptr)
{
    /* postorder tree traversal */
    if (ptr) {
        postorder(ptr->leftChild);
        postorder(ptr->rightChild);
        printf("%d", ptr->data);
    }
}
```

Program 5.3: Postorder traversal of a binary tree

(3) 실행 예

```
C:\Windows\system32\cmd.exe
the length of input string should be less than 80
input string <postfix expression> : AB/C*D*E+
creating its binary tree

inorder traversal : A/B*C*D*E
preorder traversal : ***/ABCDE
postorder traversal : AB/C*D*E+

계속하려면 아무 키나 누르십시오 . . .
```

2. 후위표현식(postfix expression)을 입력받아 Figure 5.16과 같은 이진트리를 구성한 후, 반복문을 사용한 중위순회 및 level-order 순회를 하는 프로그램을 작성하라.(9점)

(1) 실행순서

① 후위표현식으로 부터 산술식의 이진트리를 생성한다. ※ createBinTree();

※ 1번 실행순서 ①과 동일

데이터 입력형식
- 입력파일(input.txt) : AB/C*D*E+
- 피연산자(Operands) : 알파벳 한 글자
- 연산자(Operators) : +, -, *, /, %

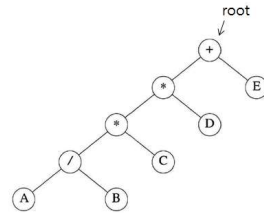


Figure 5.16: Binary tree with arithmetic expression

② 반복문을 사용한 이진트리 중위순회를 통해 데이터를 출력한다. (4점)

※ iterInorder(root); // ①에서 사용한 스택을 그대로 사용

③ level-order traversal을 통해 각 노드의 데이터를 출력한다. (5점)

※ levelOrder(root); // circular queue 사용

(2) 구현 세부사항

```
typedef struct node *treePointer;
typedef struct node {
    char data; // 문자출력
    treePointer leftChild, rightChild;
}node;
treePointer root;

// stack
treePointer stack[MAX_STACK_SIZE];
int top = -1;

// circular queue
treePointer queue[MAX_QUEUE_SIZE];
int front=0, rear=0;
```

- 데이터 출력 시 printf("%c", ptr->data); 사용
- stack과 queue의 각 항목은 treePointer 타입
- stack empty 및 queue empty는 NULL을 리턴

① 이진트리 순회

```
int top = -1; /* initialize stack */
treePointer stack[MAX_STACK_SIZE];

void iterInorder(treePointer node)
{
    for (;;) {
        for (; node; node = node->leftChild)
            push(node); /* add to stack */
        node = pop(); /* delete from stack */
        if (!node) break; /* empty stack */
        printf("%d", node->data);
        node = node->rightChild;
    }
}
```

Program 5.4: Iterative inorder traversal

```
int front = 0; int rear = 0;
treePointer queue[MAX_QUEUE_SIZE];

void levelOrder(treePointer ptr)
{ /* level order tree traversal */
    if (!ptr) return; /* empty tree */
    addq(ptr);
    for (;;) {
        ptr = deleteq();
        if (ptr) {
            printf("%d", ptr->data);
            if (ptr->leftChild)
                addq(ptr->leftChild);
            if (ptr->rightChild)
                addq(ptr->rightChild);
        }
        else break;
    }
}
```

Program 5.5: Level-order traversal of a binary tree

② 정적인 환형큐(circular queue)

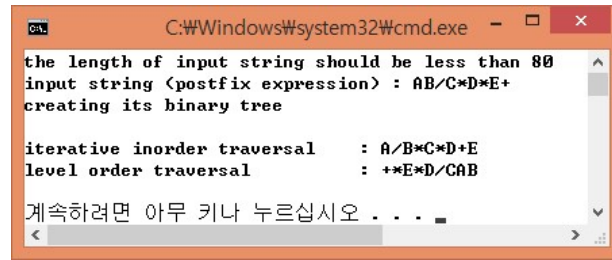
```
void addq(element item)
{ /* add an item to the queue */
    rear = (rear+1) % MAX_QUEUE_SIZE;
    if (front == rear)
        queueFull(); /* print error and exit */
    queue[rear] = item;
}
```

Program 3.7: Add to a circular queue

```
element deleteq()
{ /* remove front element from the queue */
    element item;
    if (front == rear)
        return queueEmpty(); /* return an error key */
    front = (front+1) % MAX_QUEUE_SIZE;
    return queue[front];
}
```

Program 3.8: Delete from a circular queue

(3) 실행 예



```
C:\Windows\system32\cmd.exe
the length of input string should be less than 80
input string <postfix expression> : AB/C*D*E+
creating its binary tree

iterative inorder traversal : A/B*C*D*E
level order traversal : **E*D/CAB

계속하려면 아무 키나 누르십시오 . . .
```

■ 제출 형식

- 솔루션 이름 : DS 13
- 프로젝트 이름 : 1, 2
- 각 소스파일에 주석처리
“학번 이름”
“본인은 이 소스파일을 복사 없이 직접 작성하였습니다.”
- 솔루션 정리 후 솔루션 폴더를 “학번.zip”으로 압축하여 과제 게시판에 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감 : 4월 30일(목) 자정
- 2차 마감 : 5월 1일(금) 자정(만점의 80%)