자료구조 실습 보고서

실습 9. 수식이진트리

2016 년 5 월 25 일

학번: 201404051

이름: 정 용 석

1. 실습 문제 소개

이번 실습은 3 단계 과정으로 이루어져 있는데, 첫 번째가 A~H 까지 8 개의 변수를 이용하여 나만 수식을 3 개 만들고 이에 대한 수식 이진 트리를 그린다. 그리고 실습 내용의 템플릿에 따라서 수식 이진 트리를 구현하고 마지막으로 이진 트리를 순회할 전위, 중위, 후위 순회 함수를 만들고 처음에 만들었던 수식 3 개를 각각의 순회에 맞춰 출력한다.

2. 소스 코드

***노트북에 인코딩 문제 때문에 주석을 그냥 영어로 달았습니다.**

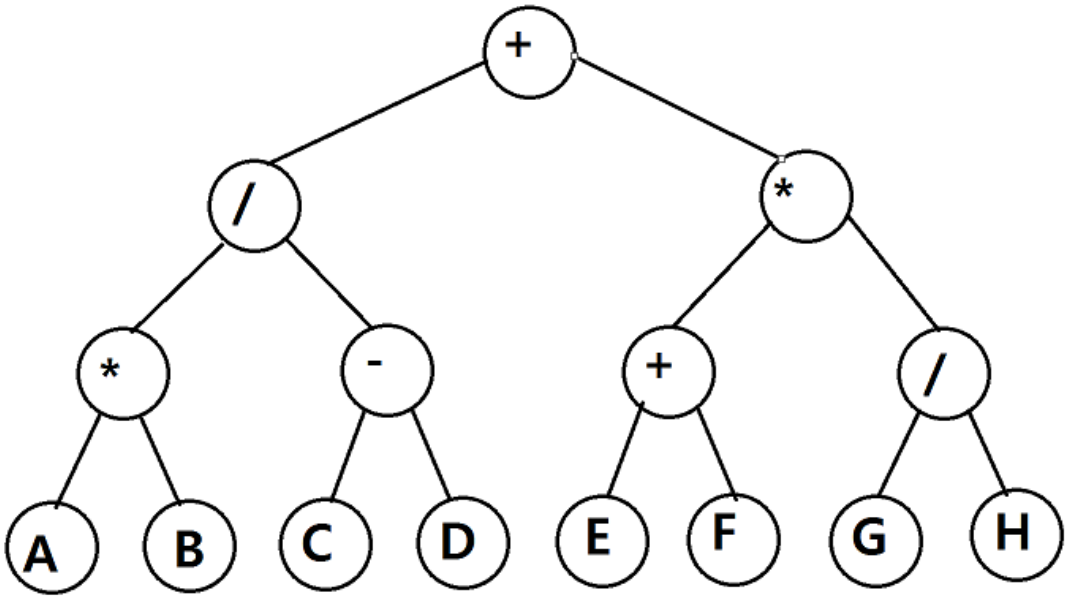
```
typedef struct _node {
        char data;
        struct _node *right;
        struct _node *left;
}node;

node *makeBT(node *left, char item, node *right) {
        node *t;
        //Make a new node T
        t = (node*)malloc(sizeof(node));
        //Connect left, right as child nodes for T
        t->left = left;
        t->right = right;
        //Item into T's data field
        t->data = item;

        return t;
}

// Pre order transverse data
void preorder(node *t)
{
        if (t == NULL)
                return;
        //order: current->left->right

        printf("%c ", t->data);
        preorder(t->left);
        preorder(t->right);
}
// In order transverse data
void inorder(node *t)
{
        if (t == NULL)
                return;
        //order: left->current->right
        inorder(t->left);
```

```
        printf("%c ", t->data);
        inorder(t->right);
}
// Post order transverse data
void postorder(node *t)
{
        if (t == NULL)
                return;
        //order: left->right->current
        postorder(t->left);
        postorder(t->right);
        printf("%c ", t->data);
}

void test1()
{
        node *a, *b, *c, *d, *e, *f, *g, *h;
        node *i, *j, *k, *l, *m, *n, *o, *root;

        a = makeBT(NULL, 'A', NULL);
        b = makeBT(NULL, 'B', NULL);
        c = makeBT(NULL, 'C', NULL);
        d = makeBT(NULL, 'D', NULL);
        e = makeBT(NULL, 'E', NULL);
        f = makeBT(NULL, 'F', NULL);
        g = makeBT(NULL, 'G', NULL);
        h = makeBT(NULL, 'H', NULL);

        i = makeBT(a, '*', b);
        j = makeBT(c, '-', d);
        k = makeBT(e, '+', f);
        l = makeBT(g, '/', h);
        m = makeBT(i, '/', j);
        n = makeBT(k, '*', l);
        root = makeBT(m, '+', n);

        printf("Expression: A * B / (C - D) + ( E + F ) * G / H ₩n₩n");

        printf("  Preorder  "); preorder(root); printf("₩n");
        printf("  Inorder  "); inorder(root); printf("₩n");
        printf("  Postorder  "); postorder(root); printf("₩n");
}
void test2()
{
        node *a, *b, *c, *d, *e, *f, *g, *h;
        node *i, *j, *k, *l, *m, *n, *root;

        a = makeBT(NULL, 'A', NULL);
        b = makeBT(NULL, 'B', NULL);
        c = makeBT(NULL, 'C', NULL);
        d = makeBT(NULL, 'D', NULL);
        e = makeBT(NULL, 'E', NULL);
```

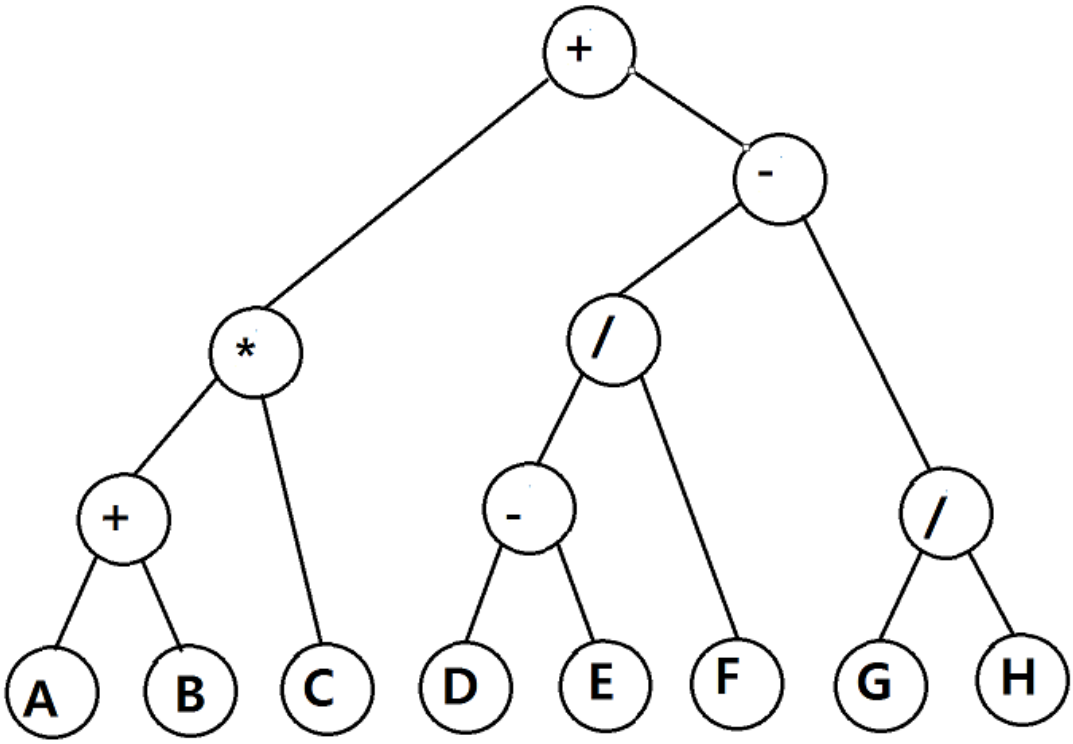
```
	f = makeBT(NULL, 'F', NULL);
	g = makeBT(NULL, 'G', NULL);
	h = makeBT(NULL, 'H', NULL);

	i = makeBT(a, '+', b);
	j = makeBT(d, '-', e);
	k = makeBT(g, '/', h);
	l = makeBT(i, '*', c);
	m = makeBT(j, '/', f);
	n = makeBT(m, '-', k);
	root = makeBT(l, '+', n);

	printf("Expression: (A + B) * C + ( D - E ) / F - ( G / H )₩n₩n");

	printf("  Preorder  "); preorder(root); printf("₩n");
	printf("  Inorder  "); inorder(root); printf("₩n");
	printf("  Postorder  "); postorder(root); printf("₩n");
}
void test3()
{
	node *a, *b, *c, *d, *e, *f, *g, *h;
	node *i, *j, *k, *l, *m, *n, *root;

	a = makeBT(NULL, 'A', NULL);
	b = makeBT(NULL, 'B', NULL);
	c = makeBT(NULL, 'C', NULL);
	d = makeBT(NULL, 'D', NULL);
	e = makeBT(NULL, 'E', NULL);
	f = makeBT(NULL, 'F', NULL);
	g = makeBT(NULL, 'G', NULL);
	h = makeBT(NULL, 'H', NULL);

	i = makeBT(b, '+', c);
	j = makeBT(e, '-', f);
	k = makeBT(i, '-', d);
	l = makeBT(j, '+', g);
	m = makeBT(a, '*', k);
	n = makeBT(l, '/', h);
	root = makeBT(m, '*', n);

	printf("Expression: A * ( ( B + C ) - D ) * ( ( E - F ) + G ) / H₩n₩n");

	printf("  Preorder  "); preorder(root); printf("₩n");
	printf("  Inorder  "); inorder(root); printf("₩n");
	printf("  Postorder  "); postorder(root); printf("₩n");
}
```

3. 테스트 결과

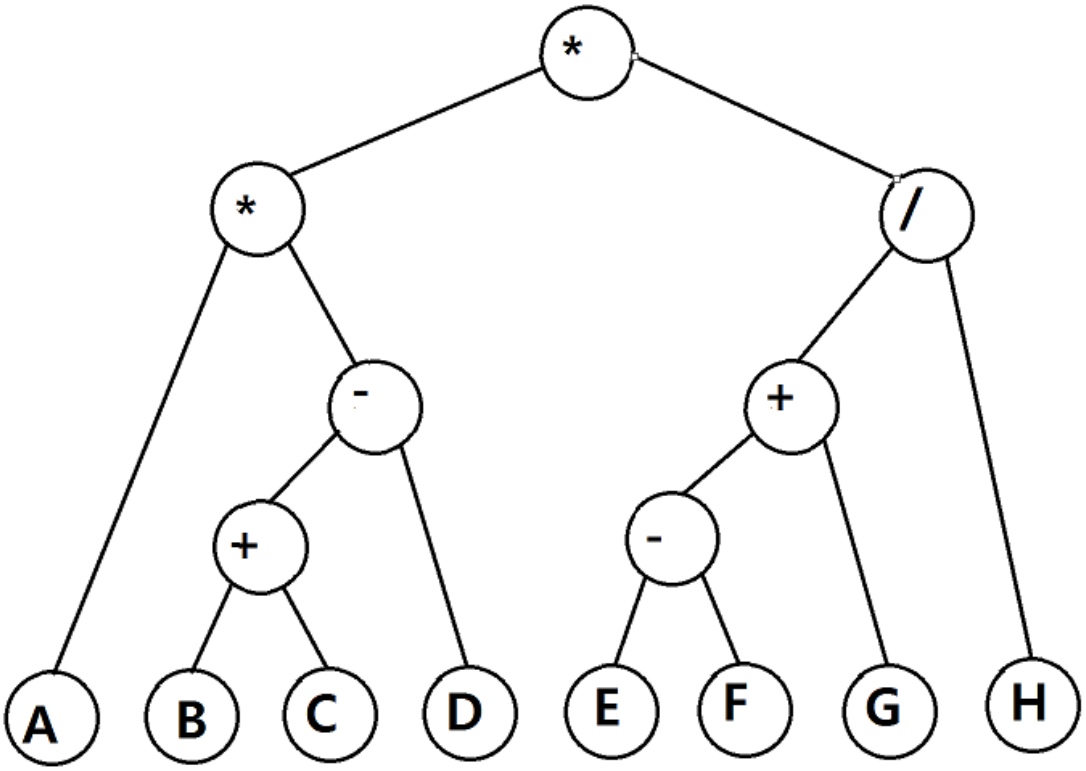
1) 수식: A * B / (C - D) + (E + F) * G / H



2) 수식: (A + B) * C + (D - E) / F - (G / H)



3) 수식: A * ((B + C) - D) * ((E - F) + G) / H




```
Expression: A * B / (C - D) + ( E + F ) * G / H

  Preorder  + / * A B - C D * + E F / G H
  Inorder  A * B / C - D + E + F * G / H
  Postorder  A B * C D - / E F + G H / * +

Expression: (A + B) * C + ( D - E ) / F - ( G / H )

  Preorder  + * + A B C - / - D E F / G H
  Inorder  A + B * C + D - E / F - G / H
  Postorder  A B + C * D E - F / G H / - +

Expression: A * ( ( B + C ) - D ) * ( ( E - F ) + G ) / H

  Preorder  * * A - + B C D / + - E F G H
  Inorder  A * B + C - D * E - F + G / H
  Postorder  A B C + D - * E F - G + H / *

Press any key to continue . . .
```

4. 작성자 코멘트

딱히 쓸 말이 없는 것 같다. 일단 이진 트리의 구현 자체는 배열도 아니고 Linked List 의 구조를 가지기 때문에 너무 쉽게 구현할 수 있다. 문제가 된다면 순회 함수 정도가 아닐까 하지만 전위, 중위, 후위 순회의 개념이 잡혀 있다면 함수 구현 또한 너무나 간단하다. 코딩 자체는 10 분도 안 걸렸던 것 같다.

개인적으로 3 개의 수식은 딱히 특정한 패턴이 있다기 보다는 그냥 8 개의 변수를 사용하고 그냥 생각나는 사칙연산을 이용하여 만들었다. 그리고 테스트 문도 단순하게 3 개를 따로 만들어서 앞에서부터 출력하는 식으로 구현하였다. 보고서에 첨부한 이진 트리 그림은 그림판을 이용하였다. 느낀 점이라고 한다면 순회 함수를 구현할 때 쓰는 재귀 함수의 편리성? 정도가 될 것 같다.