컴퓨터알고리즘과실습 실습5

2020112046 이재원

5-1,5-2

```
#include <iostream>

using namespace std;

struct node {
    char info;
    struct node *l;
    struct node *r;
};

class Stack {
   private:
    struct node *stack;
    int p;

   public:
    Stack(int max = 100) {
        stack = new node[max];
        p = 0;
    }
    ~Stack() {
        delete stack;
    }
    inline void push(node v) {
        stack[p++] = v;
    }
    inline node pop() {
        return stack[--p];
    }
    inline int empty() {
        return !p;
    }
};

void showStructure(node *p, int level) {
    int j;

    if (p != NULL) {
        showStructure(p->r, level + 1);
        for (j = 0; j < level; j++)
            cout << "\t";
        cout << p->info;
        if ((p->l != NULL) && (p->r != NULL))
            cout << "<";
        else if (p->r != NULL)
```

```
            cout << "/";
        else if (p->l != NULL)
            cout << "\\";
        cout << endl;
        showStructure(p->l, level + 1);
    }
}

void inorder(node *p) {
    if (p->l != NULL) {
        cout << "(";
        inorder(p->l);
    }
    cout << p->info;
    if (p->r != NULL) {
        inorder(p->r);
        cout << ")";
    }
}

class Queue {
   private:
    struct node *queue;
    int head, tail;

   public:
    Queue(int max = 100) {
        queue = new node[max];
        head = tail = 0;
    }
    ~Queue() {
        delete queue;
    }
    inline void push(node v) {
        queue[tail++] = v;
    }
    inline node pop() {
        return queue[head++];
    }
    inline int empty() {
        return head == tail;
    }
};

void levelorder(node *p) {
    Queue q;
    q.push(*p);
    while (!q.empty()) {
        node n = q.pop();
        cout << n.info;
        if (n.l != NULL)
            q.push(*n.l);
        if (n.r != NULL)
            q.push(*n.r);
    }
```

```
}

int main() {
    struct node *x, *root;
    char c;
    int cnt = 0;
    Stack s(50);

    // make expression tree
    while ((c = cin.get()) != '\n') {
        while (c == ' ') cin.get(c);

        x = new node;
        x->info = c;
        if (c == '+' || c == '*') {
            struct node *temp = new node;
            *temp = s.pop();
            x->l = temp;
            struct node *temp2 = new node;
            *temp2 = s.pop();
            x->r = temp2;
        }
        s.push(*x);
    }

    root = new node;
    *root = s.pop();

    // print tree
    // print(root);
    cout << endl;
    showStructure(root, 1);
    cout << endl
         << "inorder: ";
    inorder(root);
    cout << endl
         << "levelorder: ";
    levelorder(root);
}
```

실행 결과

```
❯ cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -
ABC+DE**F+*

                A
        *<
                                        B
                                +<
                                        C
                        *<
                                        D
                                *<
                                        E
                +<
                        F

inorder: ((F+((E*D)*(C+B)))*A)
levelorder: *+AF**+EDCB%
```

5-3

```
#include <iostream>

using namespace std;

struct node {
    char info;
    struct node *l;
    struct node *r;
};

class Stack {
   private:
    struct node k;
    Stack *next;

   public:
    Stack(node v) {
        this->k = v;
        this->next = NULL;
    }
    ~Stack() {
        delete next;
    }
    inline int empty() {
        return !next;
    }
    void push(node v);
    node pop();
};
Stack *head = NULL;
Stack *ptr = NULL;
```

```
void Stack::push(node v) {
    if (head == NULL) {
        head = new Stack(v);
        ptr = head;
    } else {
        Stack *temp = new Stack(v);
        temp->next = NULL;
        ptr->next = temp;
        ptr = temp;
    }
}
node Stack::pop() {
    Stack *p = ptr;
    node temp = p->k;
    ptr = head;
    if (ptr->next == NULL) {
        head = NULL;
        delete ptr;
        return temp;
    } else {
        while (ptr->next != p) {
            ptr = ptr->next;
        }
    }

    ptr->next = NULL;
    delete p;
    return temp;
}

void showStructure(node *p, int level) {
    int j;

    if (p != NULL) {
        showStructure(p->r, level + 1);
        for (j = 0; j < level; j++)
            cout << "\t";
        cout << p->info;
        if ((p->l != NULL) && (p->r != NULL))
            cout << "<";
        else if (p->r != NULL)
            cout << "/";
        else if (p->l != NULL)
            cout << "\\";
        cout << endl;
        showStructure(p->l, level + 1);
    }
}

void inorder(node *p) {
    if (p->l != NULL) {
        cout << "(";
        inorder(p->l);
    }
```

```
    cout << p->info;
    if (p->r != NULL) {
        inorder(p->r);
        cout << ")";
    }
}

class Queue {
   private:
    struct node k;
    Queue *next;

   public:
    Queue(node v) {
        this->k = v;
        this->next = NULL;
    }
    ~Queue() {
        delete next;
    }
    void push(node v);
    node pop();
};
Queue *Qhead = NULL;
Queue *Qptr = NULL;

void Queue::push(node v) {
    if (Qhead == NULL) {
        Qhead = new Queue(v);
        Qptr = Qhead;
    } else {
        Queue *temp = new Queue(v);
        temp->next = NULL;
        Qptr->next = temp;
        Qptr = Qptr->next;
    }
}

node Queue::pop() {
    Queue *p = Qhead;
    node temp = p->k;
    Qhead = Qhead->next;
    delete p;
    return temp;
}

void levelorder(node *p) {
    Qptr->push(*p);
    Qhead = Qptr;
    while (Qhead != NULL) {
        node n = Qptr->pop();
        cout << n.info << " ";
        if (n.l != NULL) {
            Qptr->push(*n.l);
        }
```

```
        if (n.r != NULL) {
            Qptr->push(*n.r);
        }
    }
}

int main() {
    struct node *x, *root;
    char c;
    int cnt = 0;

    // make expression tree
    while ((c = cin.get()) != '\n') {
        while (c == ' ') cin.get(c);

        x = new node;
        x->info = c;
        if (c == '+' || c == '*') {
            struct node *temp = new node;
            *temp = ptr->pop();
            x->l = temp;
            struct node *temp2 = new node;
            *temp2 = ptr->pop();
            x->r = temp2;
        }
        ptr->push(*x);
        // s.push(*x);
    }

    root = new node;
    *root = ptr->pop();

    // // print tree
    // // print(root);
    cout << endl;
    showStructure(root, 1);
    cout << endl
         << "inorder: ";
    inorder(root);
    cout << endl
         << "levelorder: ";
    levelorder(root);

    return 0;
}
```

실행 결과


❯ cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g test.cpp -
ABC+DE**F+*
A
*
B
+
C
*
D
*
E
+
F
inorder: ((F+((E*D)*(C+B)))*A)
levelorder: *+AF**+EDCB%