2023 Algorithm Study 1Week-스터디 시간

sookyoung0620 · 방금 전 · 비공개

2023 Algorithm Study

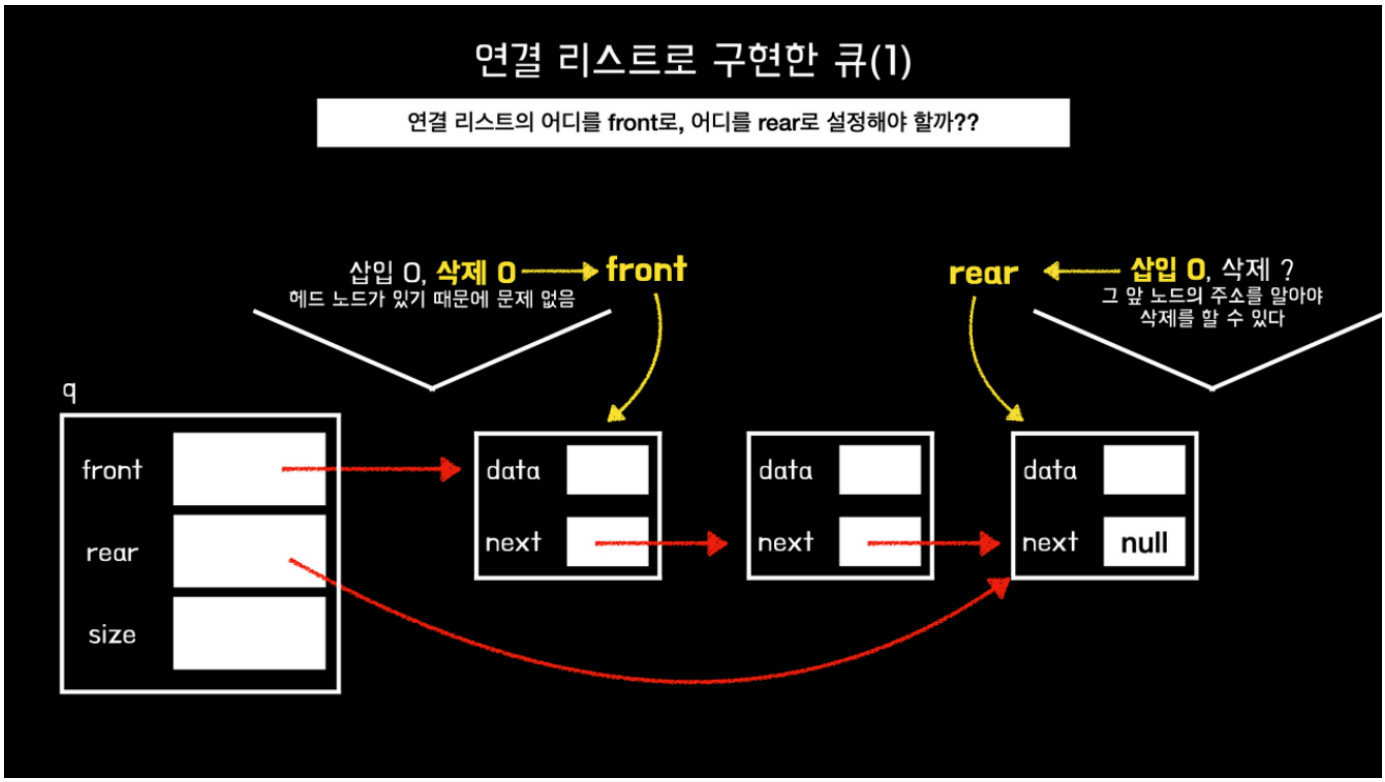
목록 보기 3/3

백준 10845번

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#pragma warning (disable:4996)

typedef struct Node {
    int data;
    struct Node* next;
} Node;

typedef struct {
    Node* front;
    Node* rear;
    int size;
} Queue;
```



> 노드 정의부터 너무 헷갈렸다.. 나의 오래전 oj를 참고해봤을 때 pre,next를 같이 만들었다
> 하지만 나 나름대로 이중연결리스트일 때 pre,next를 만들어야겠다고 이해했는데
> 너무나 혼란스러웠다.. 사실 잘 풀면 틀린 코드는 아니겠지만 나는 이 사진을 참고하여 data와 next 포인터를 만들
> 사진출처 : https://ksk9820.tistory.com/187

```
void initialize(Queue* queue) {
    queue->front = NULL;
    queue->rear = NULL;
    queue->size = 0;
}

void push(Queue* queue, int x) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = x;
    newNode->next = NULL;

    if (queue->size == 0) {
        queue->front = newNode;
        queue->rear = newNode;
    }
    else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }

    queue->size++;
}

int pop(Queue* queue) {
    if (queue->size == 0) {
```

```
            return -1;
        }

        int result = queue->front->data;
        Node* temp = queue->front;

        if (queue->size == 1) {
            queue->front = NULL;
            queue->rear = NULL;
        }
        else {
            queue->front = queue->front->next;
        }

        free(temp);
        queue->size--;

        return result;
    }

    int size(Queue* queue) {
        return queue->size;
    }

    int empty(Queue* queue) {
        if (queue->size == 0) {
            return 1;
        }
        else
            return 0;
    }

    int front(Queue* queue) {
        if (queue->size == 0) {
            return -1;
        }
        return queue->front->data;
    }

    int back(Queue* queue) {
        if (queue->size == 0) {
            return -1;
        }
        return queue->rear->data;
    }

    int main() {
        Queue queue;
        initialize(&queue);
```

> 정말 소름돋게도 또 &를 안 써서 엄청 오래걸렸다... 제발 썼으면

```
        int N;
        scanf("%d", &N);
```

> 아래 코드는 정말 많이 나오는 거 같다 외우자

```
    while (N>0) {
        char command[6];
        scanf("%s", command);

        if (strcmp(command, "push") == 0) {
            int x;
            scanf("%d", &x);
            push(&queue, x);
        }
        else if (strcmp(command, "pop") == 0) {
            printf("%d\n", pop(&queue));
        }
        else if (strcmp(command, "size") == 0) {
            printf("%d\n", size(&queue));
        }
        else if (strcmp(command, "empty") == 0) {
            printf("%d\n", empty(&queue));
        }
        else if (strcmp(command, "front") == 0) {
            printf("%d\n", front(&queue));
        }
        else if (strcmp(command, "back") == 0) {
            printf("%d\n", back(&queue));
        }
    }

    return 0;
}
```

백준 1991번: 트리 순회

```
#pragma warning(disable: 4996)
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct node { //적정이진트리
    char data;
    struct node* left;
    struct node* right;
}Node;
```

> 적정이진트리 노드를 만들어준다..

```
void freeTree(Node* tree);
void preorder(struct node* tree);
void addLeftRight(Node* tree, char x, char y, char z);
```

```
void printPreorder(struct node* node);
void printInoder(struct node* node);

int main() {
    int N;
    char x, y, z;
```

> 자료구조 1주차 oj를 많이 참고했는데 이 문제는 알파벳을 받기도 하고 비어있는 노드에서는 .을 받기 때문

```
Node* root, * tree = (struct node*)malloc(sizeof(struct node));
Node* leftchild = (struct node*)malloc(sizeof(struct node)), * rightchild = (struct node*)malloc(siz

    root = tree;
    scanf("%d", &N);
    (void)getchar();
```

> 자꾸 반환 값이 무시되었다고 떴다... 맨 처음에 getchar()를 안 써서 이거만 추가하면 해결될 줄 알았는데 안됐다.. getchar() 반환값이 무시되었다는 오류창도 있어서 (void)getchar();로 바꿔줬더니 getchar() 반환값이 무시되었다는 오류는 없어졌다

```
    for (int i = 0; i < N; i++) {
        scanf("%c", &x);
        (void)getchar();
        scanf("%c", &y);//왼쪽 자식
        (void)getchar();
        scanf("%c", &z);//오른쪽 자식
        (void)getchar();

        if (i == 0) {
            tree->data = x;
            leftchild->data = y;
            rightchild->data = z;
            leftchild->left = NULL;
            leftchild->right = NULL;
            rightchild->left = NULL;
            rightchild->right = NULL;
            if (y != '.') { //왼쪽 자식이 있는 경우 저장
                root->left = leftchild;
            }
            else {
                free(leftchild);
            }
            if (z != '.') { //오른쪽 자식이 있는 경우 저장
                root->right = rightchild;
            }
            else {
                free(rightchild);
```

```
                }
            }
            else {
                addLeftRight(root, x, y, z);
            }
        }
```

노드에 입력값을 넣어주는 과정을 동일하게 했다.. 하지만 오류가 잔뜩 있는 걸 보아하니 내가 새로 추가한

```
        printf("전위순회\n");
        printPreoder(root);
        printf("중위순회\n");
        printInoder(root);
```

```
    freeTree(root);

}

void freeTree(Node* tree) {
    if (tree == NULL) {
        return;
    }
    freeTree(tree->left);
    freeTree(tree->right);

    free(tree);  //메모리 해제
}

void preorder(struct node* tree) {
    if (tree == NULL) {
        return;
    }

    printf(" %c", tree->data);

    preorder(tree->left);
    preorder(tree->right);
}

void addLeftRight(Node* tree, char x, char y, char z) {
    if (tree == NULL) {
        return;
    }

    Node* leftchild = (Node*)malloc(sizeof(Node));
    Node* rightchild = (Node*)malloc(sizeof(Node));

    leftchild->data = y;
    rightchild->data = z;
    leftchild->left = NULL;
    leftchild->right = NULL;
    rightchild->left = NULL;
    rightchild->right = NULL;
```

```
        if (tree->data == x) {
            if (y == '.') { //왼쪽 자식이 없는 경우
                tree->left = NULL;
                free(leftchild);
            }
            else {
                tree->left = leftchild;
            }
            if (z == '.') { //오른쪽 자식이 없는 경우
                tree->right = NULL;
                free(rightchild);
            }
            else {
                tree->right = rightchild;
            }
        }
        else {
            free(leftchild);
            free(rightchild);
        }
        addLeftRight(tree->left, x, y, z);
        addLeftRight(tree->right, x, y, z);
    }

    void printPreorder(struct node* node) {
        if (node == NULL) {
            return;
        }
        printf("%c", node->data);
        printPreorder(node->left);
        printPreorder(node->right);

    }

    void printInoder(struct node* node) {
        if (node == NULL) {
            return;
        }
        printInorder(node->left);
        printf("%c", node->data);
        printInorder(node->right);
    }
```

> 처음에 data를 int로 받았었다.. 밑에서 char로 바꾼 걸 까먹고...
> 그래서 char data로 바꿔주고 출력할때도 %c를 썼지만 해결하지 못했다..

> 오류1: 반환값이 무시되었습니다.
> 오류2: NULL 포인터 역참조
> 오류3: printInrder 함수에서 참조되는 알수 없는 외부 기호
> 정말 소름돋게도 printinorder에서 r을 빼먹어서 이 무시무시한 오류가 난 거였따
> 제발 오타조심...하지만 후위순회는 출력에 오류가 있는 거 같다

최종코드

```
typedef struct node { //적정이진트리
    char data;
    struct node* left;
    struct node* right;
}Node;

void freeTree(Node* tree);
void preorder(struct node* tree);
void addLeftRight(Node* tree, char x, char y, char z);
void printPreorder(struct node* node);
void printInorder(struct node* node);
void printPostorder(struct node* node);

int main() {
    int N;
    char x, y, z;
    Node* root, * tree = (struct node*)malloc(sizeof(struct node));
    Node* leftchild = (struct node*)malloc(sizeof(struct node)), * rightchild = (struct node*)mal

    root = tree;
    scanf("%d", &N);
    (void)getchar();

    for (int i = 0; i < N; i++) {
        scanf("%c", &x);
        (void)getchar();
        scanf("%c", &y);//왼쪽 자식
        (void)getchar();
        scanf("%c", &z);//오른쪽 자식
        (void)getchar();

        if (i == 0) {
            tree->data = x;
            leftchild->data = y;
            rightchild->data = z;
            leftchild->left = NULL;
            leftchild->right = NULL;
            rightchild->left = NULL;
            rightchild->right = NULL;
            if (y != '.') { //왼쪽 자식이 있는 경우 저장
                root->left = leftchild;
            }
            else {
                free(leftchild);
            }
            if (z != '.') { //오른쪽 자식이 있는 경우 저장
                root->right = rightchild;
            }
            else {
                free(rightchild);
            }
        }
        else {
            addLeftRight(root, x, y, z);
        }
```

```
        }
        printf("전위순회\n");
        printPreorder(root);
        printf("\n중위순회\n");
        printInorder(root);
        printf("\n후위순회\n");
        printPostorder(root);

        freeTree(root);

    }

    void freeTree(Node* tree) {
        if (tree == NULL) {
            return;
        }
        freeTree(tree->left);
        freeTree(tree->right);

        free(tree);  //메모리 해제
    }

    void preorder(struct node* tree) {
        if (tree == NULL) {
            return;
        }

        printf(" %c", tree->data);

        preorder(tree->left);
        preorder(tree->right);
    }

    void addLeftRight(Node* tree, char x, char y, char z) {
        if (tree == NULL) {
            return;
        }

        Node* leftchild = (Node*)malloc(sizeof(Node));
        Node* rightchild = (Node*)malloc(sizeof(Node));

        leftchild->data = y;
        rightchild->data = z;
        leftchild->left = NULL;
        leftchild->right = NULL;
        rightchild->left = NULL;
        rightchild->right = NULL;

        if (tree->data == x) {
            if (y == '.') { //왼쪽 자식이 없는 경우
                tree->left = NULL;
                free(leftchild);
            }
            else {
                tree->left = leftchild;
            }
            if (z == '.') { //오른쪽 자식이 없는 경우
                tree->right = NULL;
                free(rightchild);
            }
            else {
                tree->right = rightchild;
```

```
            }
        }
        else {
            free(leftchild);
            free(rightchild);
        }
        addLeftRight(tree->left, x, y, z);
        addLeftRight(tree->right, x, y, z);
    }

    void printPreorder(struct node* node) {
        if (node == NULL) {
            return;
        }
        printf("%c", node->data);
        printPreorder(node->left);
        printPreorder(node->right);

    }

    void printInorder(struct node* node) {
        if (node == NULL) {
            return;
        }
        printInorder(node->left);
        printf("%c", node->data);
        printInorder(node->right);
    }

    void printPostorder(struct node* node) {
        if (node == NULL) {
            return;
        }
        printInorder(node->left);
        printInorder(node->right);
        printf("%c", node->data);
    }
```

윤수경

이전 포스트
2023 Algorithm Study 2Week

0개의 댓글

댓글을 작성하세요

댓글 작성