4190.103A-001: Programming Practice Lab. 14

Jaewon Choi
Keunsan Park
Haeun Lee
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

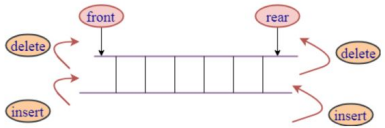
Spring 2023

지난 실습 & 과제 풀이

실습 1 - Deque

Q. Deque

Deque이란 Double Ended Queue의 준말로 기존의 큐와는 다르게 front, rear 어디로든 push & pop 이 가능한 자료 구조 형태를 의미합니다.
여러 종류의 명령어들이 입력될 때, 그에 맞는 수행을 함수들을 작성해봅시다.



- `p f 정수` : 큐의 front에 정수를 삽입합니다.
- `g f` : 큐의 front에서 데이터를 가져(삭제)옵니다.
- `p r 정수` : 큐의 rear에 정수를 삽입합니다.
- `g r` : 큐의 rear에서 데이터를 가져(삭제)옵니다.

입력

- 첫째줄에 정수 N이 주어집니다.
- 다음 N개의 줄에 명령어들이 입력됩니다.

출력

- N개의 명령어를 수행한 뒤의 큐의 상태를 출력합니다.

```c
typedef struct node{
    int data;
    struct node *next;
    struct node *prev;
}Node;

typedef struct deque{
    Node* front;
    Node* rear;
    int count;
}Deque;

Deque* dq;

void initQueue(){
    dq = (Deque*)malloc(sizeof(Deque));
    dq->front = dq->rear = NULL;
    dq->count = 0;
}

void pushFront(int n){
    Node *new = (Node*)malloc(sizeof(Node));
    new->next = new->prev = NULL;
    new->data = n;
    if(!dq->count)
        dq->front = dq->rear = new;
    else{
        new->next = dq->front;
        new->prev = dq->rear;
        dq->rear->next = new;
        dq->front->prev = new;
        dq->front = new;
    }
    dq->count += 1;
}

void pushBack(int n){
    Node *new = (Node*)malloc(sizeof(Node));
    new->next = new->prev = NULL;
    new->data = n;

    if(!dq->count)
        dq->front = dq->rear = new;
    else{
        new->next = dq->front;
        new->prev = dq->rear;
        dq->rear->next = new;
        dq->front->prev = new;
        dq->rear = new;
    }
    dq->count += 1;
}
```

```c
int getFront(){
    if(!dq->count)
        return -1;
    Node *ptr = dq->front;
    int data = ptr->data;
    dq->front = dq->front->next;
    dq->front->prev = dq->rear;
    dq->rear->next = dq->front;
    dq->count -= 1;
    free(ptr);
    return data;
}

int getBack(){
    if(!dq->count)
        return -1;
    Node *ptr = dq->rear;
    int data = ptr->data;
    dq->rear = dq->rear->prev;
    dq->rear->next = dq->front;
    dq->front->prev = dq->rear;
    dq->count -=1;
    free(ptr);
    return data;
}

void printDeque(){
    Node*ptr = dq->front;
    for(int i=0;i<dq->count;i++){
        printf("%d ",ptr->data);
        ptr = ptr->next;
    }
    printf("\n");
}

int main(){
    int N;
    initQueue();
    scanf("%d",&N);
    for(int i=0;i<N;i++){
        char pg, rf;
        int data;
        scanf(" %c %c",&pg,&rf);
        if(pg == 'p'){
            scanf("%d",&data);
            if(rf == 'r')
                pushBack(data);
            else
                pushFront(data);
        }
        else{
            if(rf == 'r')
                getBack();
            else
                getFront();
        }
    }
    printDeque();
    return 0;
}
```

실습 2 - Hashing

Q. Hashing

N명의 사람들에 대해서 2개의 Hash Table로 저장하려 합니다. 하나의 Hash Table의 key값은 이름의 맨 앞글자(A, B, C ...)이며, 다른 Hash Table의 key값은 나이대(10, 20, 30 ...)입니다.

입력

- 첫째줄에 N이 주어집니다.
- 다음 N개의 줄에 `이름 나이` 의 형태의 정보들이 주어집니다.
- 마지막줄에 `알파벳 나이대` 의 형태의 명령어가 주어집니다.

> 'A 20' 이 주어지면 이름의 맨 앞글자가 A로 시작하는 20대를 의미합니다.

출력

- 마지막 명령어에 해당하는 사람들의 이름을 모두 출력합니다.
- 해당하는 사람이 없을 시에 공백을 출력합니다.

```c
typedef struct info{
    char name[30];
    int age;
    struct info* next;
}Info;

Info *Ntable[NSLOT];
Info *Atable[ASLOT];

void initTable(){
    for(int i=0;i<NSLOT;i++)
        Ntable[i] == NULL;
    for(int i=0;i<ASLOT;i++)
        Atable[i] == NULL;
}

void insertNtable(char *name,int age){
    Info *new = (Info*)malloc(sizeof(Info));
    strcpy(new->name,name);
    new->age = age;
    new->next = NULL;
    int c = (int)(name[0] - 'A');

    if(Ntable[c] == NULL)
        Ntable[c] = new;
    else{
        Info *ptr = Ntable[c];
        while(ptr->next != NULL)
            ptr = ptr->next;
        ptr->next = new;
    }
}

void insertAtable(char *name,int age){
    Info *new = (Info*)malloc(sizeof(Info));
    strcpy(new->name,name);
    new->age = age;
    new->next = NULL;
    int c = age / 10;

    if(Atable[c] == NULL)
        Atable[c] = new;
    else{
        Info *ptr = Atable[c];
        while(ptr->next != NULL)
            ptr = ptr->next;
        ptr->next = new;
    }
}
```

```c
void printTable(){ // For debugging
    printf("NTABLE > \n");
    for(int i=0;i<NSLOT;i++){
        printf("(%c) ",i+'A');
        Info*ptr = Ntable[i];
        while(ptr != NULL){
            printf("%s ",ptr->name);
            ptr = ptr->next;
        }
        printf("\n");
    }
    printf("ATABLE > \n");
    for(int i=0;i<ASLOT;i++){
        printf("(%d) ",i);
        Info*ptr = Atable[i];
        while(ptr != NULL){
            printf("%s ",ptr->name);
            ptr = ptr->next;
        }
        printf("\n");
    }
}

int main(){
    int N, age, k2;
    char name[30], k1;
    initTable();
    scanf("%d",&N);

    for(int i=0;i<N;i++){
        scanf("%s %d",name,&age);
        insertNtable(name,age);
        insertAtable(name,age);
    }
    scanf(" %c %d",&k1,&k2);

    int nk = k1-'A', ak = k2/10;
    Info *np,*ap;

    np = Ntable[nk];
    ap = Atable[ak];

    while(np) {
        ap = Atable[ak];
        while(ap) {
            if (!strcmp(ap->name, np->name))
                printf("%s ", ap->name);

            ap = ap->next;
        }
        np = np->next;
    }
    return 0;
}
```

실습 3 - Circular Queue

Q. Circular Queue

Queue 자료구조는 FIFO(First in First Out)에 따라 데이터가 pop되는 형태입니다.
평범한 Queue의 각 노드에 중요도를 부여하고, 기존의 순서를 바꾸는 동작을 수행하려 합니다.

1. 현재 Queue의 front가 가르키는 노드의 중요도를 확인합니다.
2. 현재 노드의 중요도보다 높은 중요도를 가진 노드가 하나라도 있다면, 현재 노드를 rear로 이동시킵니다.
3. 만약 더 높은 중요도가 없다면, 현재 front가 가르켰던 node를 pop 합니다.

처음 상태의 큐에서 K번째 노드(0번째부터 시작)가 몇번째로 pop되는지 구해봅시다.

입력

- 첫째줄에 노드의 갯수 N과 K가 주어집니다.
- 둘째줄에 각 노드의 중요도가 주어집니다.

출력

- K번째 노드가 pop되는 순서를 출력합니다.

```c
#include<stdio.h>
#include<stdlib.h>

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

typedef struct queue{
    Node *front;
    Node *rear;
    int count;
}Queue;

Queue *cq;

void initQueue(){
    cq = (Queue*)malloc(sizeof(Queue));
    cq->front = cq->rear = NULL;
    cq->count = 0;
}

void insert(int n){
    Node *new = (Node*)malloc(sizeof(Node));
    new->next = NULL;
    new->data = n;

    if(!cq->count)
        cq->front = cq->rear = new;
    else{
        new->next = cq->front;
        cq->rear->next = new;
        cq->rear = new;
    }
    cq->count += 1;
}

int pop(){
    if(!cq->count)
        return -1;
    Node *ptr = cq->front;
    int data = cq->front->data;
    cq->front = cq->front->next;
    cq->rear->next = cq->front;
    free(ptr);
    cq->count -= 1;
    return data;
}
```

```c
void print(){
    Node*ptr = cq->front;
    for(int i=0;i<cq->count;i++){
        printf("%d ",ptr->data);
        ptr = ptr->next;
    }
    printf("\n");
}

int main(){
    int N,K;
    initQueue();
    scanf("%d %d",&N, &K);
    int imp[N];
    for(int i=0;i<N;i++){
        scanf("%d",imp+i);
        insert(i);
    }

    int popped = -1, cnt=0;
    while(popped != K){
        int midx = cq->front->data, max = imp[midx];
        for(int i=0;i<N;i++){
            if(!imp[i]) continue;
            if(max < imp[i]){
                max = imp[i];
                midx = i;
            }
        }
        while(cq->front->data != midx) {
            int data = pop();
            insert(data);
        }
        popped = pop();
        imp[midx] = 0;
        cnt++;
    }
    printf("%d",cnt);

    return 0;
}
```

과제 1 - 다이얼 돌리기

Q. 다이얼 돌리기

철수는 다이얼을 **양방향 순환 큐** 의 형태로 구현하였다.
큐의 크기가 N일 때 각 노드는 `1 ~ N` 의 값을 가진다.
다음의 3가지 동작만 가능할 때, 원하는 번호를 뽑기 위해 **최소** 몇번 이동해야 하는지 구해보자.

- **동작 1** : 첫번째 위치한 번호를 뽑는다.

  > 번호(1)를 뽑으면 다음 번호(2)가 첫번째가 된다.

- **동작 2** : 왼쪽으로 한 칸 돌린다.

  > '1, 2, .. 10' 에서 '2, 3 ... 10, 1'이 된다.

- **동작 3** : 오른쪽으로 한 칸 돌린다.

  > '1, 2, .. 10' 에서 '10, 1 ... 9'가 된다.

입력

- 첫째줄에 큐의 크기 N과, 몇 개의 번호를 뽑을지를 나타내는 M이 주어진다.
- 둘째줄에 뽑고자 하는 번호가 순서대로 한 줄에 주어진다.

출력

- 주어진 번호를 뽑는데 필요한 '동작 2' 와 '동작 3'의 최소 횟수를 출력한다.

```c
typedef struct node{
    int data;
    struct node* next;
    struct node* prev;
}Node;

typedef struct deque{
    Node* front;
    Node* rear;
    int count;
}Deque;

Deque* dq;

void initQueue(){
    dq = (Deque*)malloc(sizeof(Deque));
    dq->front = dq->rear = NULL;
    dq->count = 0;
}

void pushFront(int n){
    Node *new = (Node*)malloc(sizeof(Node));
    new->next = new->prev = NULL;
    new->data = n;
    if(!dq->count)
        dq->front = dq->rear = new;
    else{
        new->next = dq->front;
        new->prev = dq->rear;
        dq->rear->next = new;
        dq->front->prev = new;
        dq->front = new;
    }
    dq->count += 1;
}

void pushBack(int n){
    Node *new = (Node*)malloc(sizeof(Node));
    new->next = new->prev = NULL;
    new->data = n;

    if(!dq->count)
        dq->front = dq->rear = new;
    else{
        new->next = dq->front;
        new->prev = dq->rear;
        dq->rear->next = new;
        dq->front->prev = new;
        dq->rear = new;
    }
    dq->count += 1;
}
```

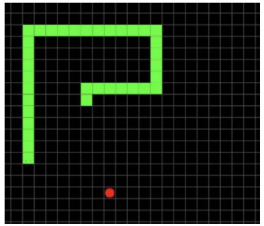
```c
int popFront(){
    if(!dq->count)
        return -1;
    Node *ptr = dq->front;
    int data = ptr->data;
    dq->front = dq->front->next;
    dq->front->prev = dq->rear;
    dq->rear->next = dq->front;
    dq->count -= 1;
    free(ptr);
    return data;
}

int popBack(){
    if(!dq->count)
        return -1;
    Node *ptr = dq->rear;
    int data = ptr->data;
    dq->rear = dq->rear->prev;
    dq->rear->next = dq->front;
    dq->front->prev = dq->rear;
    dq->count -=1;
    free(ptr);
    return data;
}

int rotate(int want){
    int left=0,right=0;
    Node *ptr = dq->front;
    while(ptr->data != want){
        ptr = ptr->next;
        right++;
    }
    ptr = dq->front;
    while(ptr->data != want){
        ptr = ptr->prev;
        left++;
    }
    if(left > right){
        for(int i=0;i<right;i++){
            int data = popFront();
            if(data < 0) return -1;
            pushBack(data);
        }
        return right;
    }
    else{
        for(int i=0;i<left;i++){
            int data = popBack();
            if(data < 0) return -1;
            pushFront(data);
        }
        return left;
    }
}
```

```c
int main(){
    int N,M;
    int total=0;
    initQueue();

    scanf("%d %d",&N, &M);
    for(int i=0;i<N;i++)
        pushBack(i+1);

    for(int i=0;i<M;i++){
        int want,cur = dq->front->data;
        scanf("%d",&want);
        if(cur != want){
            int cnt = rotate(want);
            if(cnt < 0)
                printf("Error\n");
            else
                total += cnt;
        }
        popFront();
    }
    printf("%d",total);

    return 0;
}
```

과제 2

Q. Dummy

Dummy 라는 게임은 뱀이 나와서 기어다니며 사과를 먹는 게임이다. 사과를 먹으면 뱀 길이가 늘어나고, 기어다니다가 벽 또는 자기 자신과 부딪히면 게임은 종료된다.



NxN 정사각 보드위에서 진행되며, 몇 개의 칸에 사과가 놓여져 있다. 정사각 보드의 상하좌우 끝엔 벽이 있다. 게임이 시작할 때 뱀은 x, y 좌표로 (1, 1)에 위치하고 길이는 1이다. 뱀은 처음에 오른쪽을 향한다.
뱀은 매 초마다 이동하는데, 다음의 규칙을 따른다.

1. 뱀은 몸길이를 늘려 머리를 다음칸에 위치시킨다.
2. 벽이나 자기자신의 몸에 부딪히면 게임은 종료된다.
3. 이동한 칸에 사과가 있다면, 그 칸의 사과는 사라지고 꼬리의 위치는 그대로 있게 된다.
4. 이동한 칸에 사과가 없다면, 몸 길이가 줄어 꼬리가 위치했던 칸이 비워진다. 즉 몸의 길이는 변하지 않는다.

위 규칙에 대해서, 사과의 위치와 뱀의 이동경로가 주어질 때 게임이 종료될때까지 몇 초가 걸리는지 출력해보자.

입력

- 첫째줄에 N 값이 주어진다. (2 <= N <= 100)
- 다음줄에 사과의 개수 K가 주어진다.
- 다음 K개의 줄에 사과의 위치가 두 정수(행과 열) 로 주어진다.
- 그 다음 줄에 L 이 주어진다.
- 다음 L개의 줄에 방향 전환 정보가 주어진다.

> '정수 알파벳' 의 형태로 주어지며, 게임 시작으로 X(정수)초 뒤에 왼쪽(L) 또는 오른쪽(D)으로 90도 회전하라는 명령어들이 주어진다.

출력

- 게임이 종료된 시간(초)를 출력한다.

```c
int map[101][101] = {0,}; //사과 위치
int exist[101][101] = {0, }; //뱀이 있는 공간
int dir[2][4] = {{0, 1, 0, -1}, {1, 0, -1, 0}}; // 회전 방향 (0번째 행 : y 좌표, 1번째 행 : x 좌표)

typedef struct Node{
    struct Node *next;
    int y;
    int x;
}Node;

typedef struct Queue{
    Node *front;
    Node *rear;
    int count;
}Queue;

void init(Queue *q){
    q->front = q->rear = NULL;
    q->count = 0;
}

int isEmpty(Queue *q){
    return q->count == 0;
}

void push(Queue *q, int y, int x){
    Node *new = (Node*)malloc(sizeof(Node));
    new->next = NULL;
    new->y = y;
    new->x = x;
    exist[y][x] += 1;

    if(isEmpty(q))
        q->front = new;
    else
        q->rear->next = new;
    q->rear = new;
    q->count += 1;
}

void pop(Queue *q){
    Node *ptr = q->front;
    q->front = ptr->next;
    exist[ptr->y][ptr->x] = 0;
    q->count -=1;
    free(ptr);
}
```

```c
int main(){
    int N,K,L,X;
    Queue queue;
    init(&queue);

    scanf("%d", &N);
    scanf("%d", &K);
    for(int i=0;i<K;i++){ // 사과 위치 저장
        int x, y;
        scanf("%d %d", &y, &x);
        map[y][x] = 1;
    }

    scanf("%d", &L);
    int info[L][2];

    for(int i=0;i<L;i++){ // 방향 전환 정보 저장
        char d;
        scanf("%d %c", &info[i][0], &d);
        if(d == 'D')
            info[i][1] = 1; // 오른쪽을 뜻함
        else
            info[i][1] = 0; // 왼쪽을 뜻함
    }

    int head = 0; // dir 배열의 0번째 열에 있는 값(0, 1)이 오른쪽을 뜻함
    int time=0, info_idx=0;
    int x=1,y=1;
    push(&queue, y, x);

    while(1){
    /* Write your code here*/
        if(time == info[info_idx][0] && info_idx < L){
            if(info[info_idx][1]) // 오른쪽
                head  = (head + 1) % 4;
            else{ // 왼쪽
                if(!head)
                    head = 3;
                else
                    head -= 1;
            }
            info_idx++;
        }
        // 이동
        y += dir[0][head];
        x += dir[1][head];
        // 저장
        push(&queue, y, x);
        // 종료 확인
        if(exist[y][x] > 1) // 겹치면
            break;
        if(y < 1 || y > N || x < 1 || x > N) // 벗어나면
            break;
        // 사과 확인
        if(map[y][x])
            map[y][x] = 0;
        else
            pop(&queue);
        time++;
    }

    printf("%d",time+1);

    return 0;
}
```

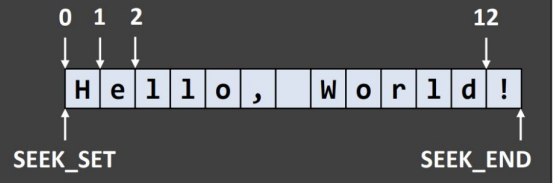
실습 & 과제

Tip

- 실습 1 : 수업 슬라이드 참고

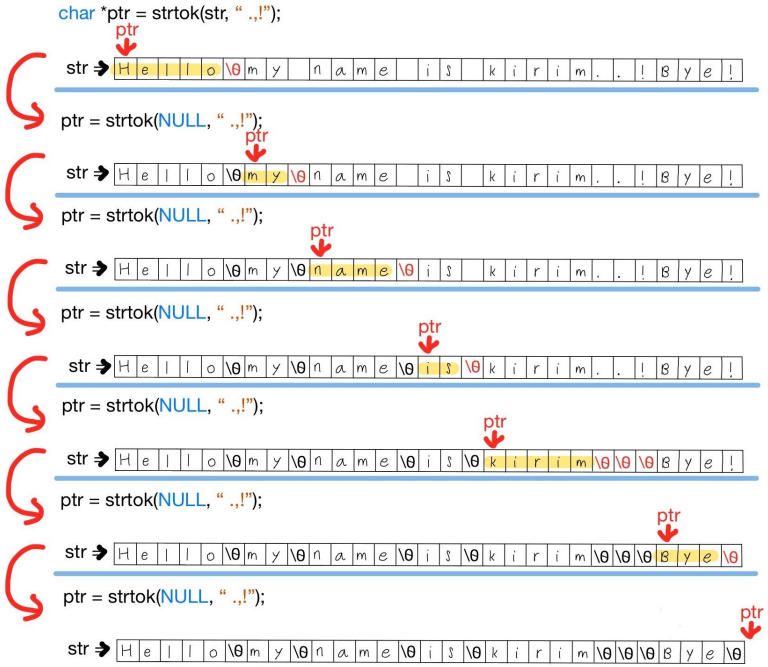
Example: Writing a File Backwards

```c
#include <stdio.h>

void main(void) {
    int c;
    FILE *fp;

    fp = fopen("in.txt", "r");
    fseek(fp, 0, SEEK_END);
    if (ftell(fp) > 0) {
        fseek(fp, -1, SEEK_CUR);
        while (1) {
            c = getc(fp);
            putchar(c);
            if (ftell(fp) > 1) fseek(fp, -2, SEEK_CUR);
            else break;
        }
    }
    fclose(fp);
}
```



Tip

- 실습 3, 과제 2 : strtok 함수
  - https://kirkim.github.io/c/2021/02/13/strtok_split.html



Tip

- 과제 1 : strtok_r 함수
  - strtok 함수랑 비슷, 하지만 여러 개의 문자열에 대해 parsing 가능
  - 아래처럼 ppStr에 현재 parsing하고 있는 문자에 대한 context를 저장하고, str 문자열을 parsing 한다.
  - https://cboard.net/s/strtok_r

```c
#include<stdio.h>
#include<string.h>
#define DELIM "-"

int main(void)
{
        char str[] = {"123-456-789"};
        char* pStr = NULL;
        char* ppStr = NULL;

        pStr = strtok_r(str, DELIM, &ppStr);

        while(pStr)
        {
                printf("pStr = ");
                puts(pStr);
                printf("ppStr = ");
                puts(ppStr);
                pStr = strtok_r(NULL, DELIM, &ppStr);
        }

        return 0;
}
```

실습 / 과제

- 과제 내용
  - 당일 진행한 실습과 관련된 문제
- 제출 기한
  - 실습 : 당일 자정까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
  - 과제 : 다음주 수요일 23:59 까지, 다음날 자정까지 지각제출 허용 (단, 점수 -30%)
- 제출 방법
  - Elice의 Submit 기능 활용