

C프로그래밍 (CSE2035) (실습20)



Ji-Hwan Kim, Ph.D.

Dept. of Computer Science and Engineering

Sogang University

Seoul, Korea

Tel: +82-2-705-8924

Email : kimjihwan@sogang.ac.kr



제출 형식

1. 각 문제에 대한 소스 코드를 압축하여 사이버캠퍼스에 업로드
 - 압축 파일명: "[실습#]학번_이름.zip" (#은 실습번호)
 - 각 소스코드 파일명: "cp실습번호_학번_p문제번호.c"
2. 업로드 후 제출 시, 제출 기한 내에 사이버캠퍼스에 제출
 - 화요일 실습의 경우: 이번주 수요일 오후 11시 59분까지
 - 목요일 실습의 경우: 이번주 금요일 오후 11시 59분까지
3. COPY 등의 문제 발생 시 실습 0점 및 각종 불이익을 줄 것



Practice 1

- 수업시간에 배운 간단한 Queue를 직접 구현하고 Dequeue 함수를 추가하여 Queue의 노드를 삭제하는 기능을 구현한다.
- 자료 구조는 Queue 강의 자료에 있는 내용을 사용한다.
- 함수의 프로토 타입은 다음과 같다.

- `void Exit(QUEUE*);`
- `void Enqueue(QUEUE *);`
- `void PrintAll(QUEUE);`
- `void Dequeue(QUEUE*);`

```
typedef struct node
{
    int          data;
    struct node* next;
} QUEUE_NODE;
```

```
typedef struct
{
    QUEUE_NODE* front;
    int         count;
    QUEUE_NODE* rear;
} QUEUE;
```

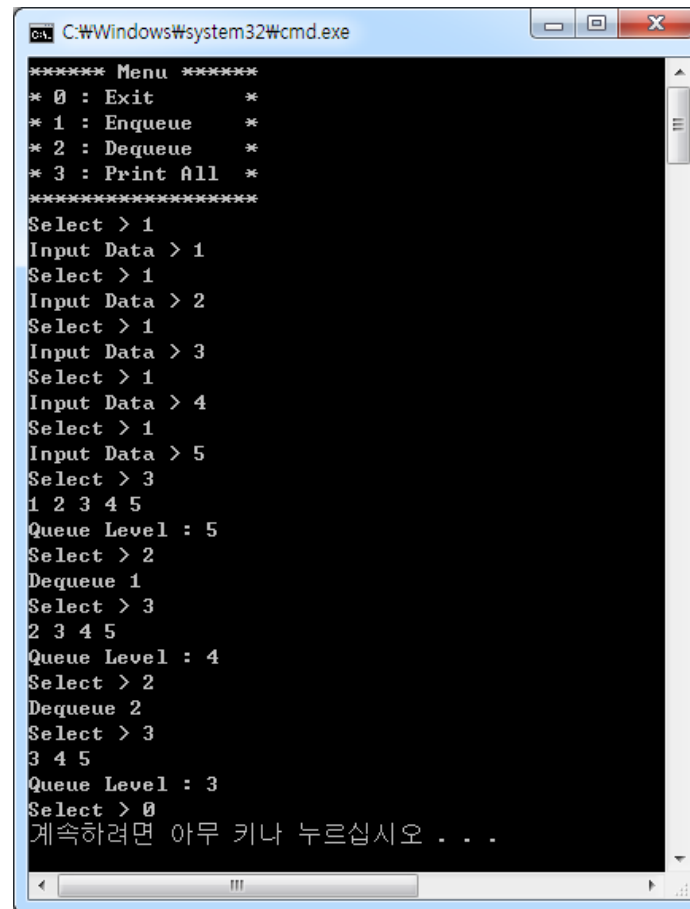


Practice 1

- 조건
- 1) 전역 변수 사용하지 말 것
- 2) **queue**는 강의 자료에서 설명한 방법으로 구현해야 한다. 다른 방법으로 구현하는 경우에 대해서는 점수 없음
- 3) **queue**의 원소를 제거할 때와 프로그램이 종료할 때 반드시 동적 할당을 한 변수/구조체들은 **free**를 해주어야 한다.
- 4) 입출력 양식 지키지 않는 경우에 대해서는 감점 (실행 결과 참고할 것)

Practice 1

- 프로그램 실행 결과



```
C:\Windows\system32\cmd.exe

***** Menu *****
* 0 : Exit          *
* 1 : Enqueue       *
* 2 : Dequeue       *
* 3 : Print All     *
*****

Select > 1
Input Data > 1
Select > 1
Input Data > 2
Select > 1
Input Data > 3
Select > 1
Input Data > 4
Select > 1
Input Data > 5
Select > 3
1 2 3 4 5
Queue Level : 5
Select > 2
Dequeue 1
Select > 3
2 3 4 5
Queue Level : 4
Select > 2
Dequeue 2
Select > 3
3 4 5
Queue Level : 3
Select > 0
계속하려면 아무 키나 누르십시오 . . .
```



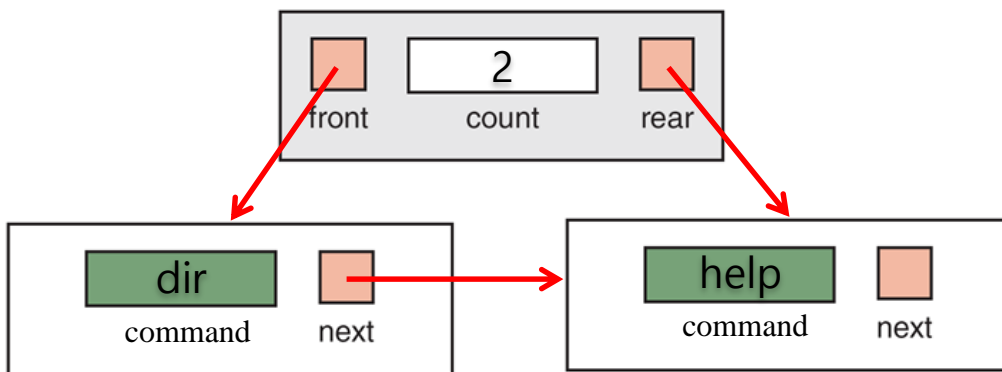
Practice 2

- **Linked list**로 구현한 **queue**에 **console** 명령어를 받아 저장하고 **“history”** 또는 **“h”** 명령어가 입력되면, 이전의 “history” 또는 “h”가 입력된 시점부터 현재까지 queue에 들어있는 명령어들을 순서대로 출력하는 프로그램을 만든다.
- **Console 명령어의 종류**
 - 이 프로그램에서는 “help”, “dir”, “mkdir”, “cd”, “history”, “h”, “quit”, “q” 으로 총 8가지의 명령어만 유효 명령어로 간주한다.
 - “history” or “h” : 현재까지의 유효한 명령어를 전부 입력된 순서대로 출력한다.
 - “quit” or “q” : 프로그램을 종료한다. (반드시 Queue에 있는 노드들은 free되어야 한다)
 - 위의 두 개 외의 유효한 명령어 6개에 대해서는 해당 명령어가 유효하다는 표시로 copy하여 출력만 해주도록 한다. (예시 참고할 것)
- 8가지 명령어 이외의 명령어의 경우는 무시한다(Queue에 넣지 않음). 이때 [Invalid]를 출력해야 한다. (예제의 find 명령어 들어왔을 때의 경우를 확인할 것)

Practice 2

■ Console Command Enqueue의 구조

- 기본적으로 수업시간에 배운 queue의 구조를 따르지만 data부분의 경우는 command를 받아야 하므로 string을 받을 수 있도록 한다.
- Head Structure인 QUEUE구조를 통해 linked queue를 관리하고, QUEUE의 count에는 현재 queue에 존재하는 node의 수를 항상 가지고 있도록 한다.



```

typedef struct node {
    char command[10];
    struct node* next;
} QUEUE_NODE;

typedef struct {
    QUEUE_NODE* front;
    int count;
    QUEUE_NODE* rear;
} QUEUE;
    
```

Practice 2

■ int CheckCommand(char* command)

- 유효한 command인지 check하는 함수
- Input: command string
- Return: 유효한 명령어인 경우에는 1, 아닌 경우에는 0을 return

■ void EnqueueCommand(Queue* pQueue, char* dataIn)

- 유효한 command를 queue에 삽입하는 함수(enqueue함수를 참고)
- Input: linked queue를 가리키는 포인터 (pQueue), 삽입할 command(dataIn)
- 결과적으로 dataIn을 가진 node가 추가된 queue가 생성되어야 한다.

■ Main function

```
int main(void) {
    char command[10];
    Queue* pQueue;
    pQueue = (Queue*)malloc(sizeof(Queue));

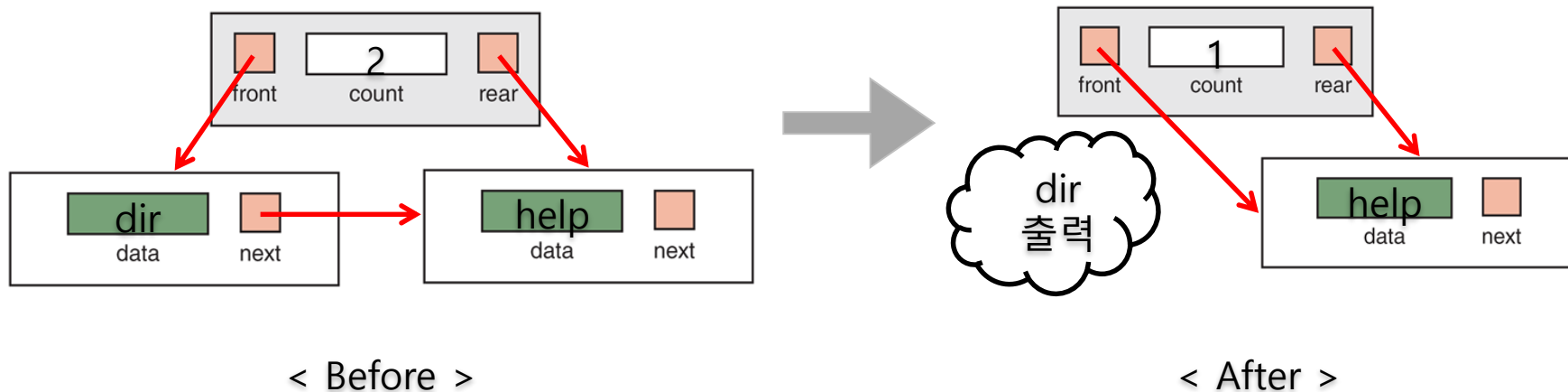
    pQueue->front = NULL;
    pQueue->count = 0;
    pQueue->rear = NULL;

    while (1) {
        //TODO(Students)
    }
    return 0;
}
```


Practice 2

■ Dequeue를 이용한 명령어 출력 방법

- Queue의 구조는 지난 실습의 구조와 같다.
- 유효한 명령어가 들어온 경우에만 queue에 삽입을 해준다.
- “history” 또는 “h” 명령어가 입력되면 queue의 저장되어 있는 모든 명령어를 dequeue operation을 통해 node하나씩 빼내고 해당 node의 data(command)를 출력해주면 된다.





Practice 2

■ `int DequeuePrint(QUEUE* pQueue, char* dataOut)`

- Queue에 들어있는 모든 command를 순서대로 빼내는 함수(dequeue함수를 참고)
- Input: linked queue를 가리키는 포인터(pQueue), 꺼낸 node의 command string(dataOut)
- Return: queue에 원소가 없을 때까지 node를 꺼내온다. 즉, return 1인 경우는 queue에 node가 남아있는 것이고, return 0인 경우는 queue가 비어있는 경우이다.
- 결과적으로 dataOut은 queue의 맨 앞 node의 command인 string을 받아온다.
- Return 값이 0일 때까지 queue에서 node의 command string을 받아온다.

■ Main function

- Queue에서 node를 꺼낼 때, 해당 node의 command string은 항상 main에서 출력한다.
- “history” 또는 “h” 명령어 입력 시에 queue에 들어있는 모든 명령어를 출력하도록 한다

.



Practice 2

- 조건
- 1) 전역 변수 사용하지 말 것
- 2) **queue**는 강의 자료에서 설명한 방법으로 구현해야 한다. 다른 방법으로 구현하는 경우에 대해서는 점수 없음
- 3) **queue**의 원소를 제거할 때와 프로그램이 종료할 때 반드시 동적 할당을 한 변수/구조체들은 **free**를 해주어야 한다.
- 4) 입출력 양식 지키지 않는 경우에 대해서는 감점 (실행 결과 참고할 것)

Practice 2

■ 프로그램 실행 결과

```
~/Documents/cprog/24$ ./a.out
>>help
[Valid] help
>>mkdir
[Valid] mkdir
>>dir
[Valid] dir
>>copy
[Invalid]
>>history
help
mkdir
dir
>>find
[Invalid]
>>quit
~/Documents/cprog/24$ ./a.out
```

```
~/Documents/cprog/24$ ./a.out
>>cd
[Valid] cd
>>h
cd
>>mkdir
[Valid] mkdir
>>help
[Valid] help
>>cp
[Invalid]
>>dir
[Valid] dir
>>q
~/Documents/cprog/24$
```