EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

제 1기

2021. 10. 17

손표훈

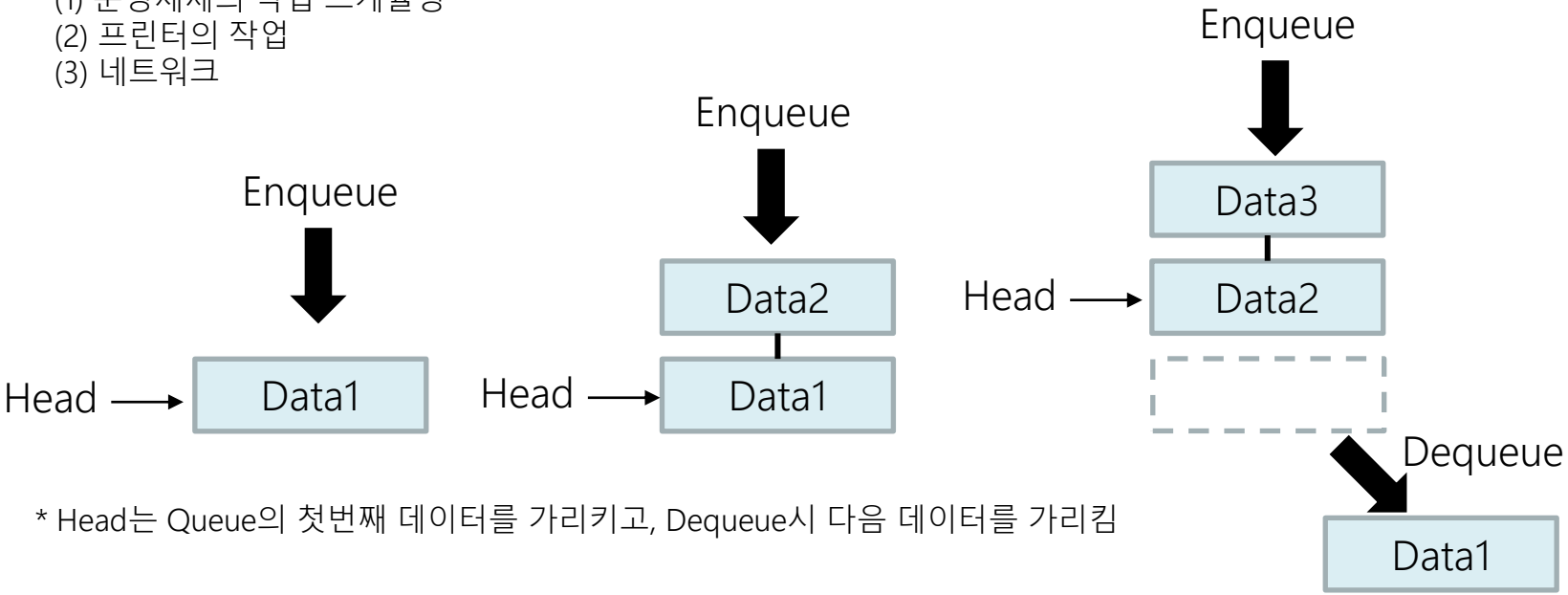
CONTENTS

EDDI
Electronic Design
Development Institute

Queue란?

※ Queue란?

- Queue는 "선입선출"의 구조(FIFO : First In First Out)를 가진 "선형" 자료구조이다.
- Queue는 Queue 프레임의 데이터를 넣는 "Enqueue"와 데이터를 빼내는 "Dequeue" 동작이 있다.
- Queue의 활용예
  (1) 운영체제의 작업 스케쥴링
  (2) 프린터의 작업
  (3) 네트워크



* Head는 Queue의 첫번째 데이터를 가리키고, Dequeue시 다음 데이터를 가리킴

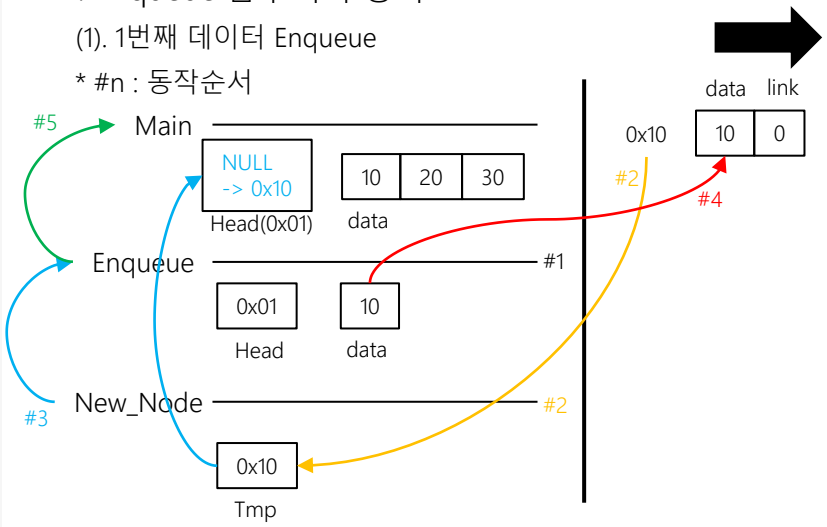
Queue의 Enqueue

1. Enqueue 함수의 기능
(1). Queue의 새 노드를 위한 메모리 할당
(2). Queue에 데이터 넣기
(3). 노드끼리 연결

2. Enqueue 함수의 추상화

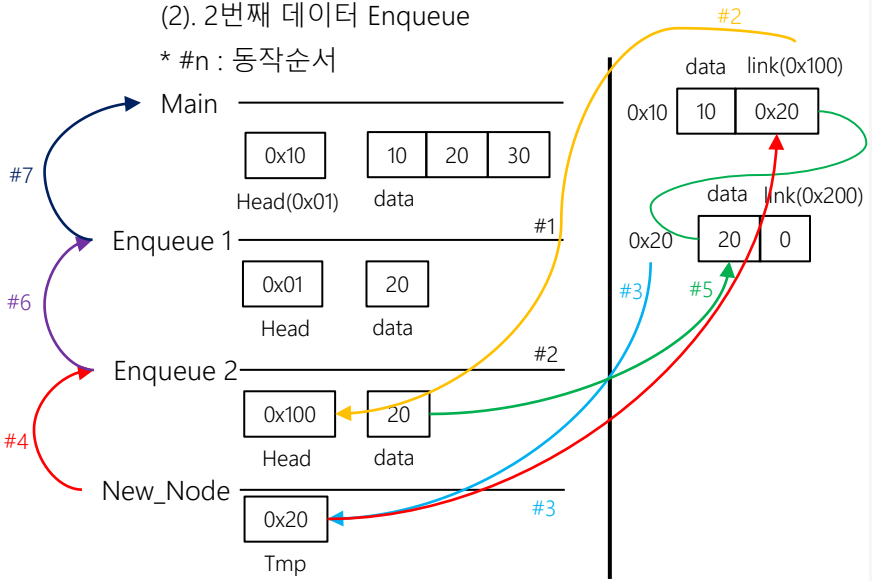
(1). 1번째 데이터 Enqueue

* #n : 동작순서



(2). 2번째 데이터 Enqueue

* #n : 동작순서



Queue의 Enqueue

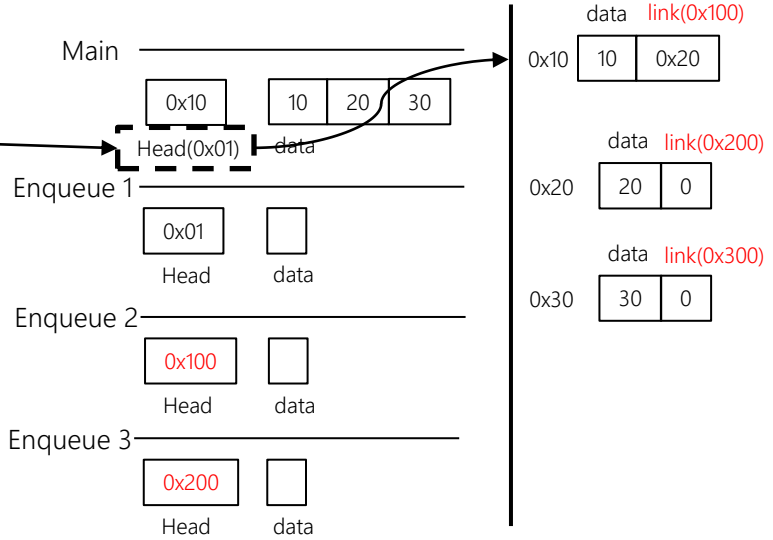
EDDI
Electronic Design Development Institute

3. Enqueue 함수 C코드 구현(재귀 호출방식)

```
queue *create_queue_node(void)
{
        queue *tmp;

        tmp = (queue *)malloc(sizeof(queue));
        tmp->link = 0;

        return tmp;
}
void enqueue_data(queue **head, int data)
{
        if (!(*head))
        {
                *head = create_queue_node();
                (*head)->data = data;
                return;
        }

        // 요기서 뭘 해야할까요 ?
        enqueue_data(&(*head)->link, data);
}
```

```
struct _queue
{
        int data;
        struct _queue *link;
};
```

* 이중포인터를 통해 Queue의 주소 값에 접근하여 Queue 프레임의 데이터 값을 변경한다.



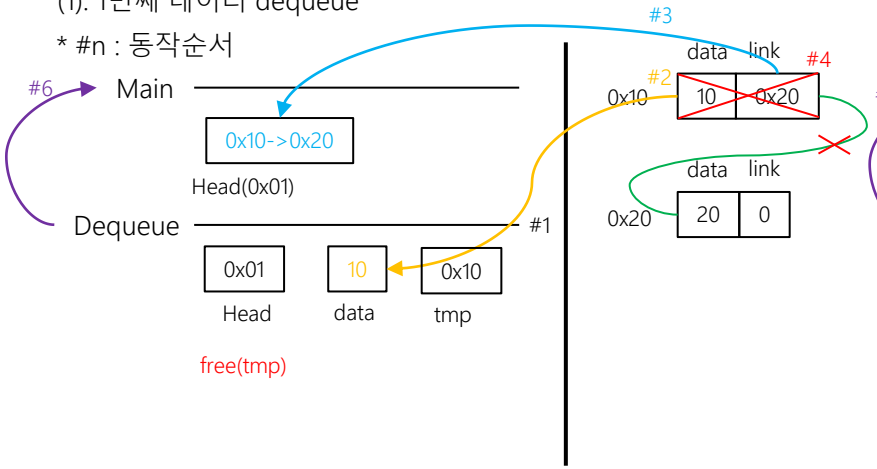
Queue의 Dequeue

1. Dequeue 함수의 기능
(1). 현재 head가 가리키는 queue공간의 데이터를 꺼낸다
(2). head가 다음 queue공간을 가리키도록 한다.
(3). 데이터가 빠진 공간의 queue의 메모리를 해제한다.
(4). queue공간이 비어 있는지 식별

2. dequeue 함수의 추상화

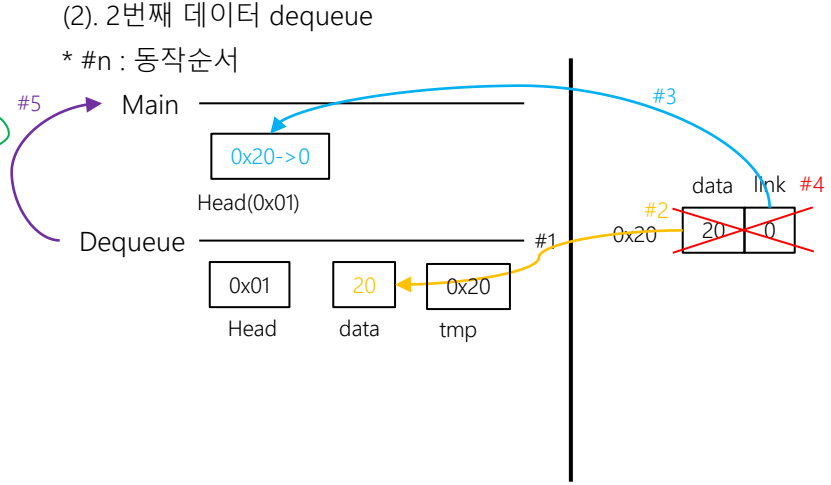
(1). 1번째 데이터 dequeue

* #n : 동작순서



(2). 2번째 데이터 dequeue

* #n : 동작순서



Queue의 Dequeue

3. dequeue 함수 C코드 구현

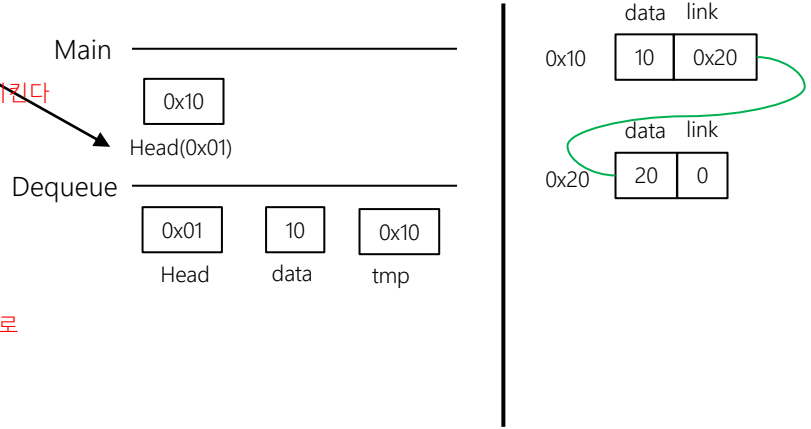
```
void dequeue_data(queue **head)
{
        if (*head)
        {
                queue *tmp = *head;
                printf("dequeue data = %d\n", (*head)->data);
                *head = (*head)->link;
                free(tmp);
        }
        else
        {
                printf("Queue is empty!!!\n");
        }
}
```

버퍼에 현재 head의 "값"을 저장하여야 한다.
저장하지 않고 head를 그대로 사용하면, free시 다음 queue 공간마저 메모리 해제가 된다.

다음 queue 공간을 head로 가리킨다

할당된 heap의 메모리공간 해제(queue공간 삭제)

head의 "값"이 0이 되면 다음 할당된 queue공간이 없다는 의미로
queue가 비어 있음을 알린다.

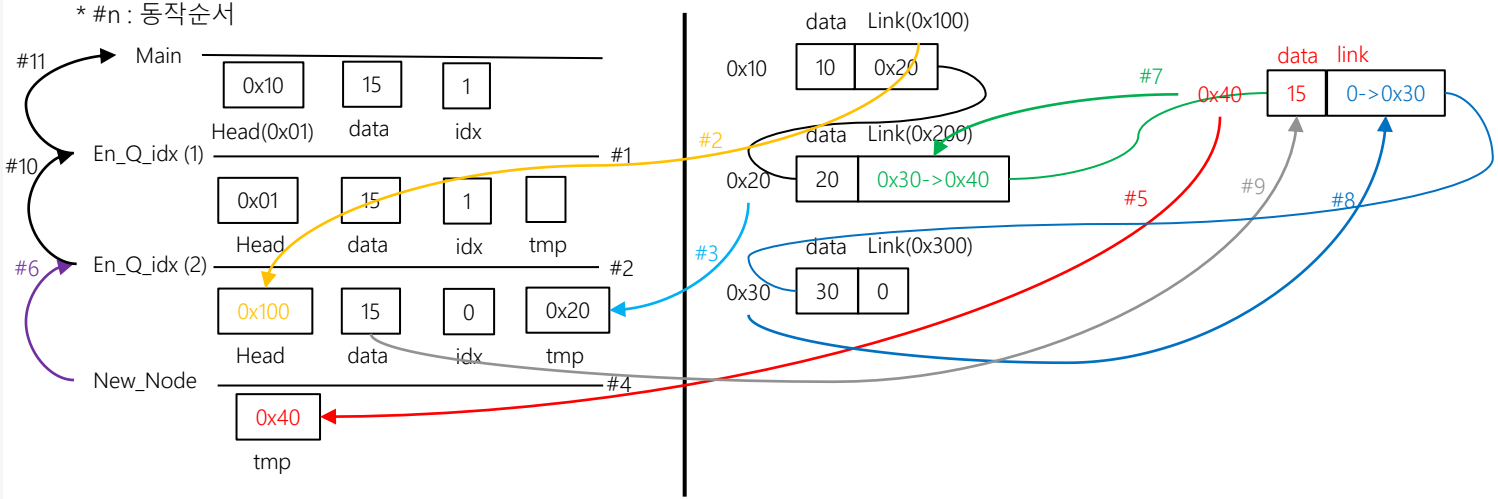


Queue의 특정 index의 Enqueue

1. Queue의 특정 index의 Enqueue 함수의 기능
(1). 특정 index값을 파라메터로 받는다.
(2). Index값과 일치하는 queue의 위치를 찾는다.
(3). Index값이 queue의 위치와 일치하면 해당 노드의 link정보를 백업한다.
(4). 해당 노드의 link에 새 노드를 생성한다.
(5). 새 노드의 link정보에 백업된 노드의 link정보를 넣고, 새 노드 앞의 노드에 새 노드 link정보를 넣는다
(6). 새 노드에 넣고자 하는 data를 넣는다
(7). 현재 index값이 queue의 사이즈보다 크거나 1개이면 예외처리

2. Enqueue_data_idx 함수의 추상화
(1). Enqueue_data_idx : 0x10과 0x20 중간에 데이터 삽입

* #n : 동작순서

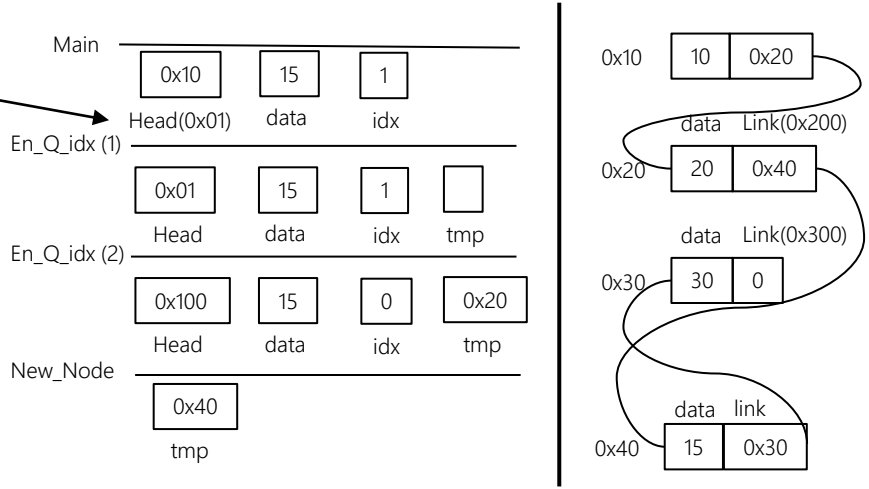


Queue의 특정 index의 Enqueue

3. enqueue_data_idx 함수 C코드 구현(재귀 호출방식)

```c
void enqueue_data_idx(queue **head, int data, int idx)
{
        // 중간에 넣는 경우 idx에 해당하는 노드 찾았을 때 새노드 추가와 link재연결
        if (!idx)
        {
                queue *tmp = create_queue_node();

                tmp->data = data;
                tmp->link = *head;

                *head = tmp;
                return;
        }

        // 예외 핸들링
        if (!(*head))
        {
                printf("작업이 불가합니다!\n");
                return;
        }

        enqueue_data_idx(&(*head)->link, data, --idx);
}
```

idx값에 해당하는 노드를 찾는 코드



Queue의 특정 index의 Enqueue

4. enqueue_data_idx 함수 C코드 구현(재귀 호출방식) – 문제점 : idx의 후위연산, 전위 연산

```
enqueue_data_idx(&(*head)->link, data, --idx);
```

*#n : 숫자가 낮을수록 최근 생성된 stack frame임

<idx의 전위 연산 사용시>

<idx의 후위 연산 사용시>

→ 좌측 전위 연산사용시 stack상태 확인결과 idx값이 3->2->1->0으로 감소하는 것을 볼 수 있으나, 후위 연산 사용시 2->3->3->3...으로 감소하지 않고 다시 3으로 돌아가 무한루프가 형성됨을 확인

→ 전위 연산의 경우 다른 변수(함수 파라메터로 값을 전달할 때 포함)에 할당할 때 할당 전 연산을 하여 바뀐 값이 할당되지만

→ 후위 연산의 경우 할당 이후에 연산을 하므로 idx값이 계속 3으로 할당되어 무한루프 형상하게 되어 오동작을 발생시킴

Queue의 특정 index의 Enqueue

EDDI
Electronic Design Development Institute

3. enqueue_data_idx 함수 C코드 구현(비재귀 호출방식)

```
void nr_enqueue_data_idx(queue **head, int data, int idx)
{
        queue *loop = *head;
        queue *backup = NULL;

        // 숏컷으로 예외처리 대신함
        while (loop && idx)
        {
                idx--;
                backup = loop;
                loop = loop->link;
        }

        if (!idx)
        {
                queue *tmp = create_queue_node();

                tmp->data = data;
                tmp->link = loop;

                if (!backup)
                {
                        *head = tmp;
                }
                else
                {
                        backup->link = tmp;
                }
                return;
        }
        printf("처리가 불가능한 작업입니다!!!\n");
```

```
// 예외 핸들링
if (!(*head))
{
        printf("작업이 불가합니다!\n");
        return;
}
```

→ 숏컷 예외처리 : AND연산자의 경우 &&앞의
피연산자가 0인 경우 뒤쪽 피연산자의 값에 상관 없이
조건이 맞지 않으므로 루프를 빠져나오게 된다.
논리 회로 결과로 불필요한 추가연산을 줄여 실행속도를 높인다.

→loop값이 연산자 앞에 위치하면서 전달받은 *head의 값이 0이면,
queue 스택프레임이 없는 경우로
While문을 실행하지 않고 넘어가게된다.

1. Queue의 특정 index의 Enqueue 함수의 기능(비재귀)

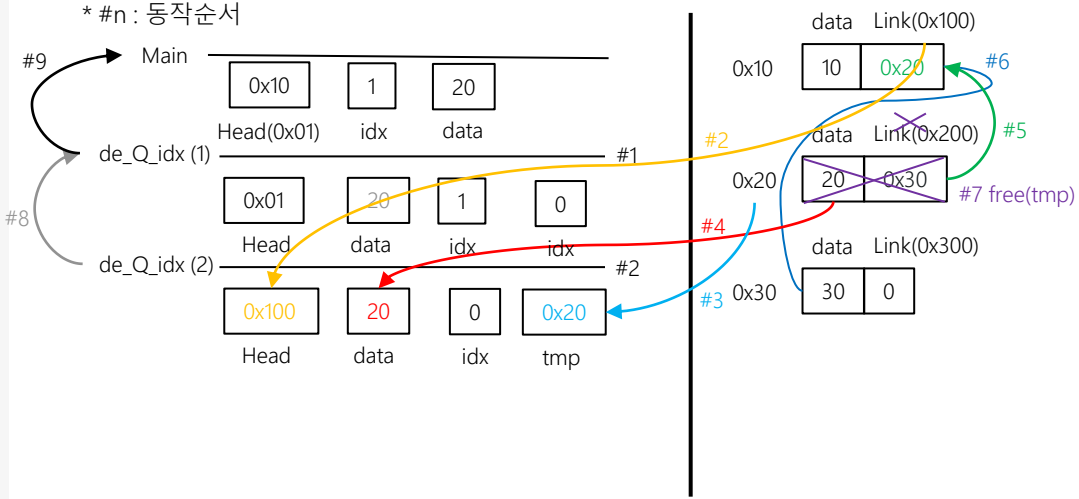
(1). 특정 index값을 파라메터로 받는다.
(2). Index값과 일치하는 queue의 위치를 찾는다.
(3). Index값이 queue의 위치와 일치하면 해당 노드의 link정보를 백업한다.
(4). 해당 노드의 link에 새 노드를 생성한다.
(5). 새 노드의 link정보에 백업된 노드의 link정보를 넣고, 새 노드 앞의 노드에 새 노드 link정보를 넣는다
(6). 새 노드에 넣고자 하는 data를 넣는다
(7). 현재 index값이 queue의 프레임이 없고, idx값이 0이 아닌경우 예외처리
(8). idx값이 queue의 사이즈 보다 큰 경우 마지막 프레임의 다음 새 노드에 추가

Queue의 특정 index의 Dequeue

1. Queue의 특정 index의 Dequeue 함수의 기능
(1). 특정 index값을 파라메터로 받는다.
(2). Index값과 일치하는 queue의 위치를 찾는다.
(3). Index값이 queue의 위치와 일치하면 해당 노드의 data정보를 백업한다.
(4). 해당 노드의 link값을 이전 노드의 link에 넣는다.
(5). 해당 노드의 메모리를 해제한다.
(7). 현재 index값이 queue의 사이즈보다 크거나 1개이면 예외처리

2. Dequeue_data_idx 함수의 추상화
(1). dequeue_data_idx : 0x20의 데이터 추출

* #n : 동작순서



Queue의 특정 index의 Dequeue

3. dequeue_data_idx 함수 C코드 구현(재귀 호출방식)

```
int dequeue_data_idx(queue **head, int idx)
{
        int data;

        //index값이 Queue의 사이즈보다 클 때
        if(!(*head))
        {
                printf("Index is over than queue size\n");
                return -1;
        }
        //Queue 중간의 데이터를 뺄 때
        if(!idx)
        {
                queue *tmp = *head;
                data = tmp->data;
                *head = tmp->link;
                free(tmp);
                return data;
        }
        return dequeue_data_idx(&(*head)->link, --idx);
}
```

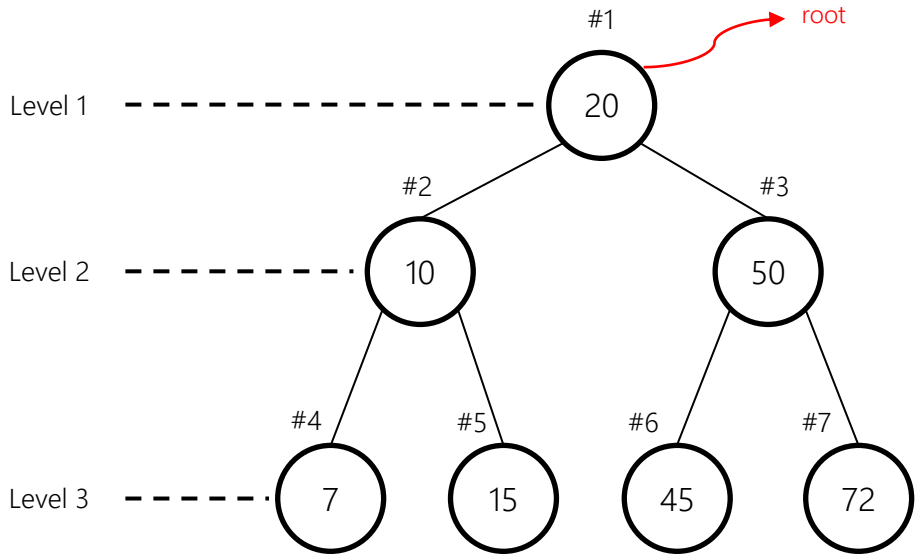
1. Queue의 특정 index의 Dequeue 함수의 기능

(1). 특정 index값을 파라메터로 받는다.
(2). Index값과 일치하는 queue의 위치를 찾는다.
(3). Index값이 queue의 위치와 일치하면 해당 노드의 data정보를 백업한다.
(4). 해당 노드의 link값을 이전 노드의 link에 넣는다.
(5). 해당 노드의 메모리를 해제한다.
(7). 현재 index값이 queue의 사이즈보다 크거나 1개이면 예외처리

Binary Tree

EDDI
Electronic Design Development Institute

1. Binary Tree의 추상화

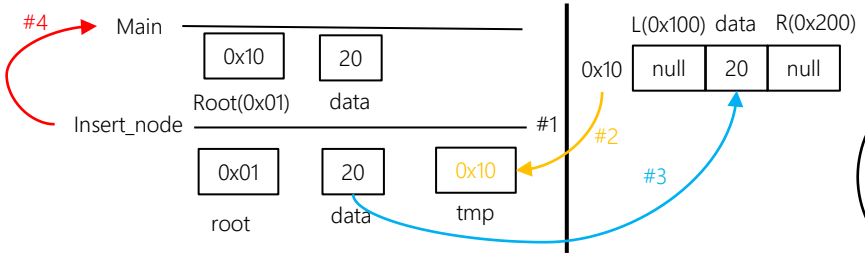


(1) 트리의 높이는 3이고, 크기는 7이다
(2) 높이는 루트부터 하위 노드단의 개수이고, 크기는 노드의 총 개수이다.

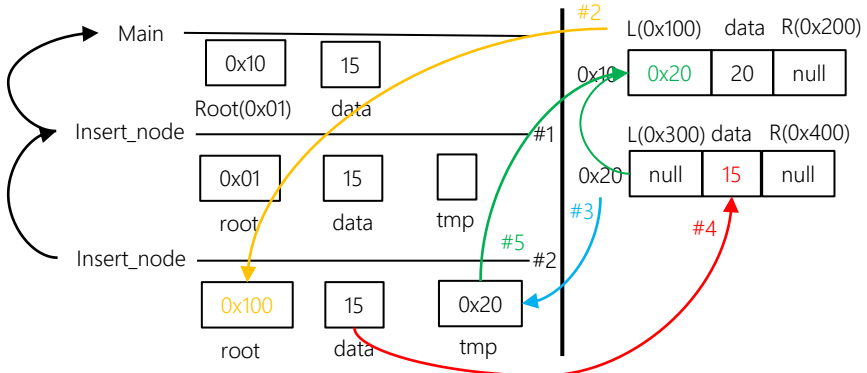
Binary Tree

2. Binary Tree의 노드 삽입 함수

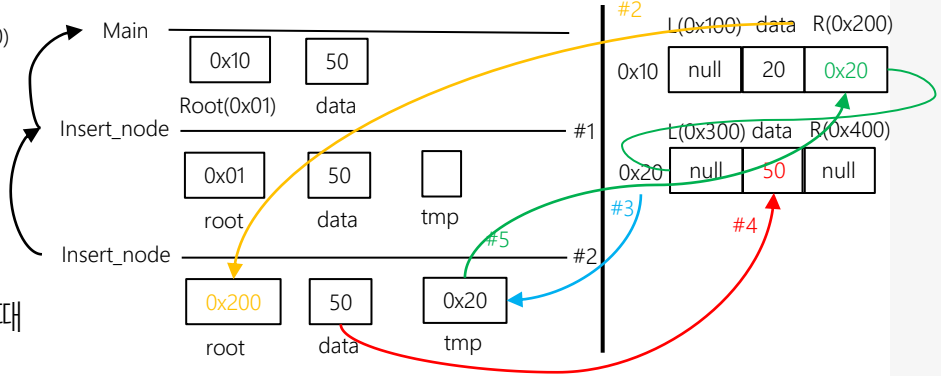
(1) Root노드가 없을 때



(2) Level 2의 노드추가, Root노드의 데이터보다 작을 때



(3) Level 2의 노드추가, Root노드의 데이터보다 클 때



※ 노드 삽입의 기능

(1). Root노드의 존재여부 확인
(2). Root노드가 없다면, 생성하고 데이터를 삽입
(3). 현재 트리의 레벨 파악
(4). Root노드가 있으면, root노드의 데이터와 데이터의 크기 비교
(5). 크면 R에 새 노드 추가, 작으면 L에 새 노드 추가
(6). 새 노드에 데이터 삽입
(8). 트리 레벨 변수 증가
(7). 현재 레벨의 노드 수가 다 존재하는지 파악