EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정
[자료구조 프로그래밍 – Queue.c]

제 1기

2021. 10. 29

박태인

목차 및 main문과 결과 화면

EDDI
Electronic Design Development Institute

1) Queue란
2) Queue의 enqueue
3) Queue의 dequeue
4) 중간 끼워 넣기 idx
5) 중간 삭제 idx
6) 재귀 호출 없는 nr_enqueue

```c
int main(void)
{
	int i;
	queue *head = NULL;
	int data[] = { 10, 20, 30, 40 };

	for (i = 0; i < 4; i++)
	{
		enqueue_data(&head, data[i]);
	}
		print_queue(head);


// #if 0 는 주석 넣는 방법 중 하나

	for (i = 0; i < 5; i++)
	{
		dequeue_data(&head);
	}

	for (i = 0; i < 4; i++)
	{
		enqueue_data(&head, data[i]);
	}

	// 특정 인덱스에 값 넣기
	enqueue_data_idx(&head, 33, 2);



#if 0

	// 특정 인덱스 값 빼기
	dequeue_data_idx(&head, 0);

	print_queue(head);
#endif

	return 0;
}
```

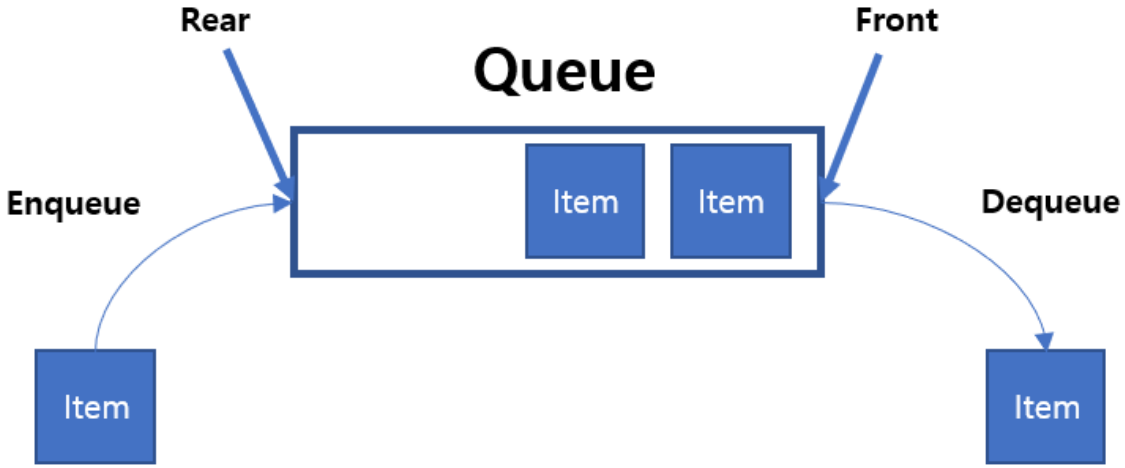
Main 문

```
queue head = 10
dequeue data = 10
dequeue data = 20
dequeue data = 30
dequeue data = 40
Queue is already empty!
```

결과 창

Queue 란?

◆ 큐는 한쪽 끝(rear)에서는 삽입 연산만 이루어지며, 다른 한 쪽 끝(front)에서는 삭제 연산만 이루어지는 유한 순서 리스트 이다.

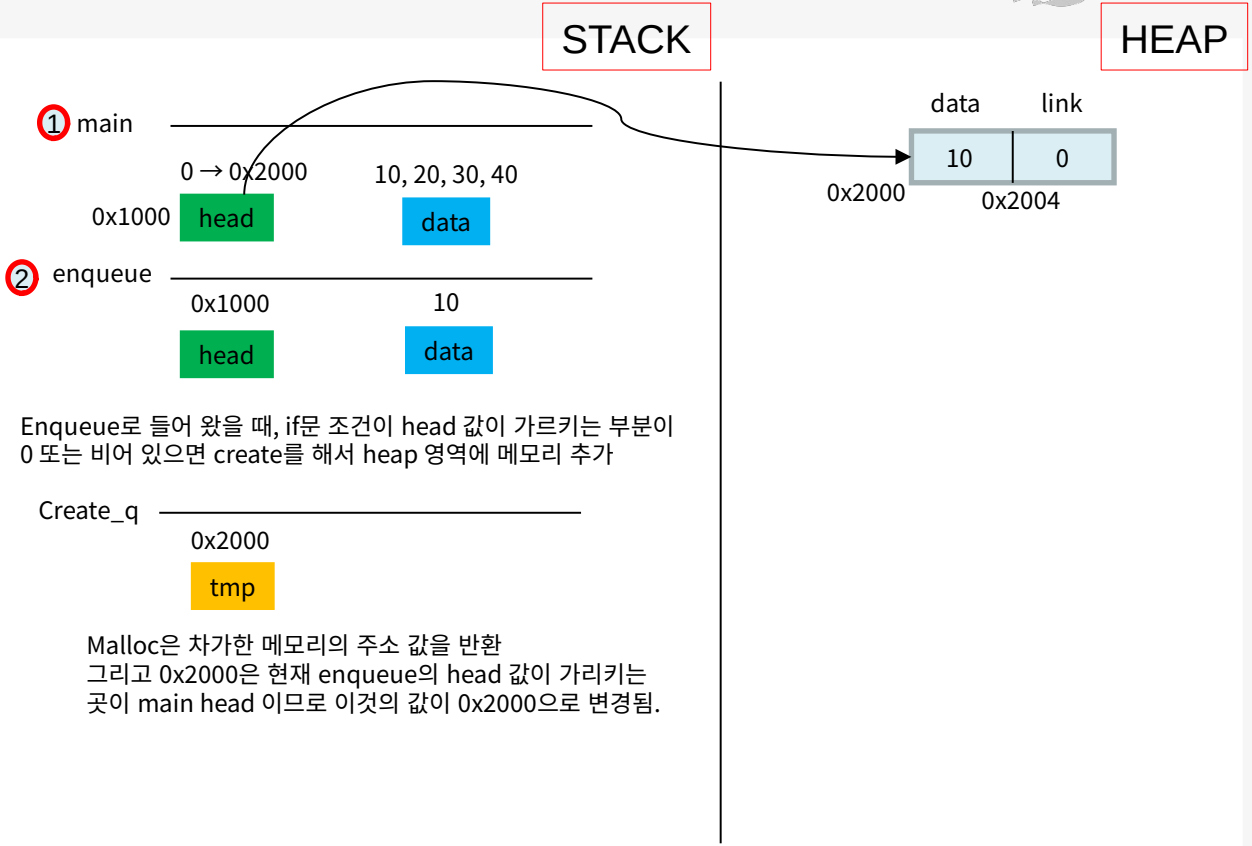


◆ 특성 : 구조상 먼저 삽입된 item이 먼저 삭제가 이루어 진다. (FIFO)

Queue 의 enqueue (1)

```c
typedef struct _queue queue;
struct _queue
{
        int data;
        struct _queue *link;
};

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

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

        return tmp;
}

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

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

```c
int main(void)
{
        int i;
    1   queue *head = NULL;
        int data[] = { 10, 20, 30, 40 };

        for (i = 0; i < 4; i++)
        {
            2   enqueue_data(&head, data[i]);
        }
                print_queue(head);
```



Enqueue로 들어 왔을 때, if문 조건이 head 값이 가르키는 부분이
0 또는 비어 있으면 create를 해서 heap 영역에 메모리 추가

Create_q

0x2000

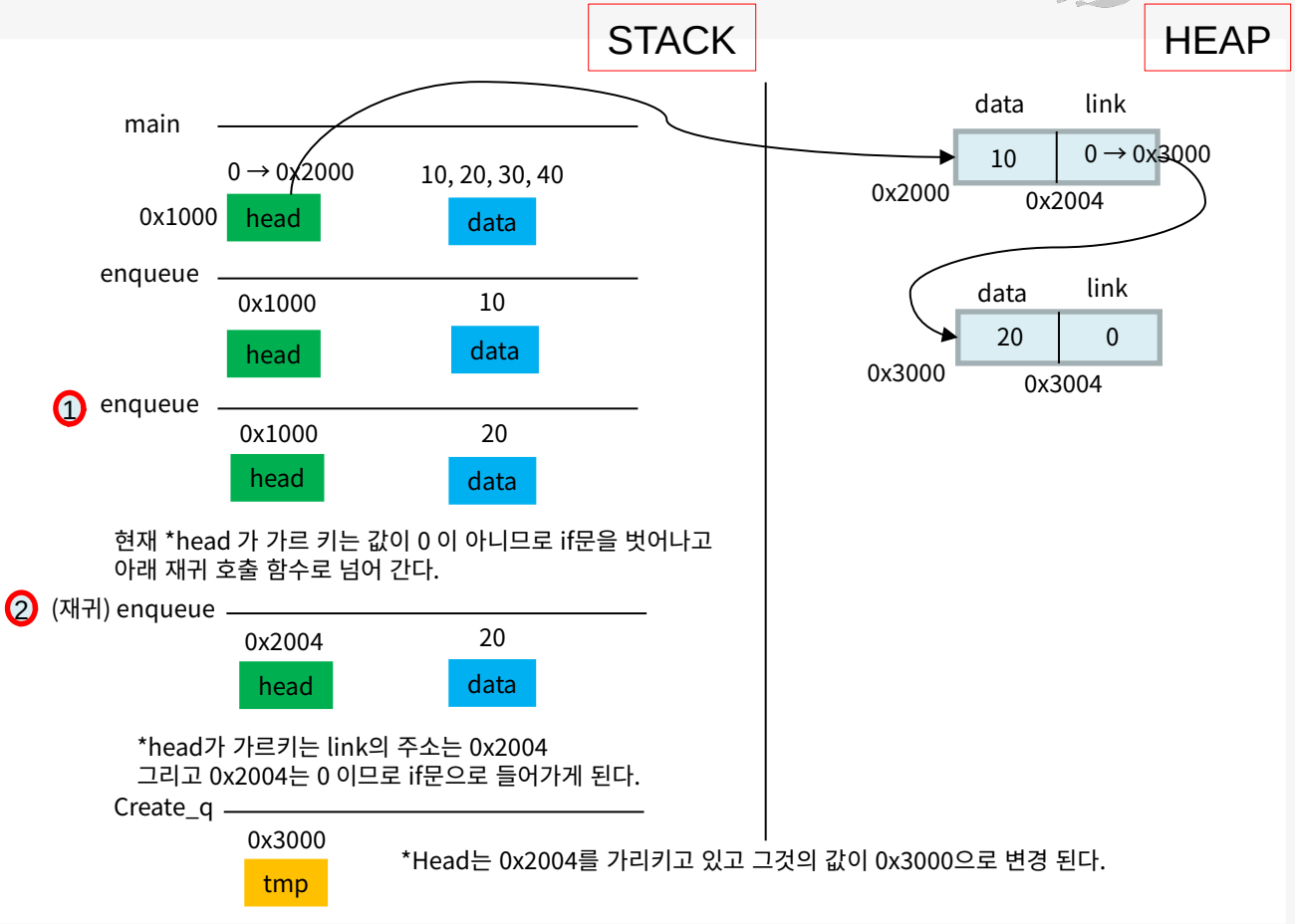
tmp

Malloc은 차가한 메모리의 주소 값을 반환
그리고 0x2000은 현재 enqueue의 head 값이 가리키는
곳이 main head 이므로 이것의 값이 0x2000으로 변경됨.

Queue 의 enqueue (2)

```c
typedef struct _queue queue;
struct _queue
{
        int data;
        struct _queue *link;
};

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 ?
        // 재귀호출
(2)     enqueue_data(&(*head)->link, data);
}
```

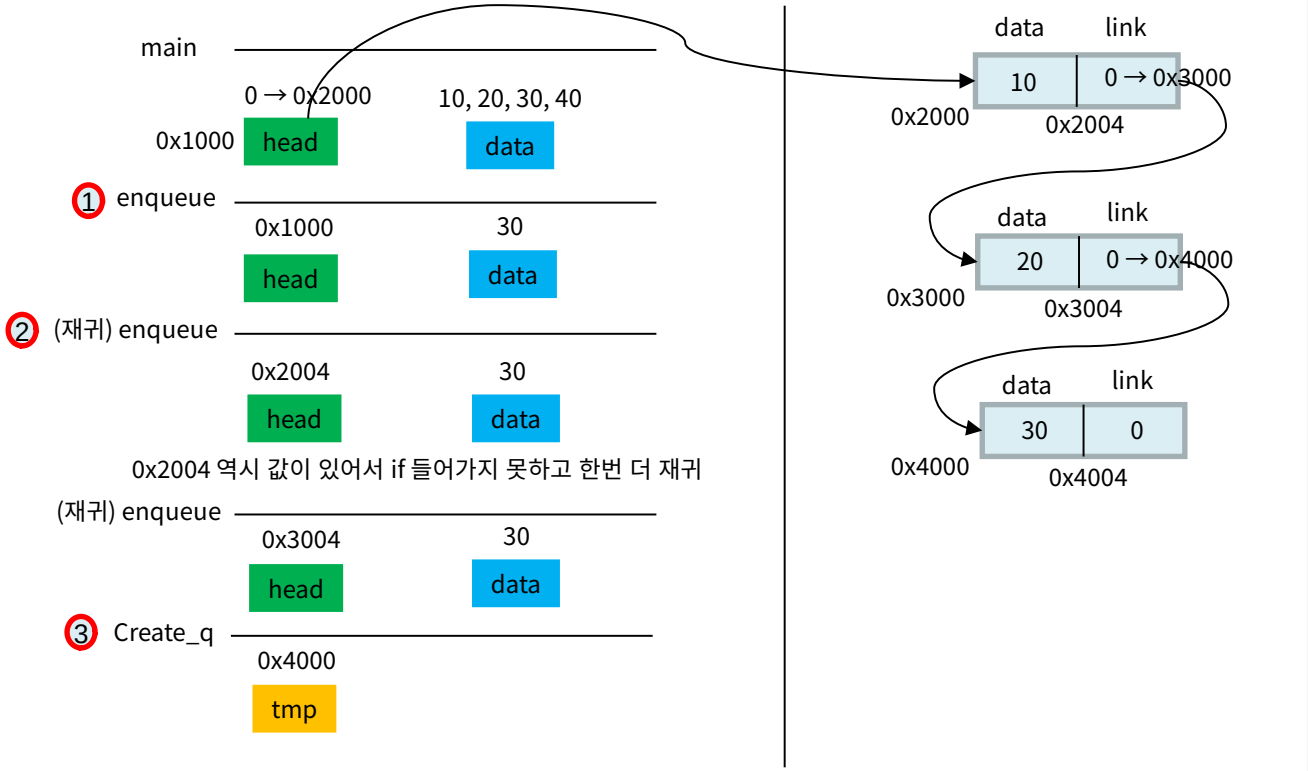
```c
int main(void)
{
        int i;
        queue *head = NULL;
        int data[] = { 10, 20, 30, 40 };

        for (i = 0; i < 4; i++)
        {
(1)             enqueue_data(&head, data[i]);
        }
                print_queue(head);
```



Queue 의 enqueue (3)

```c
typedef struct _queue queue;
struct _queue
{
        int data;
        struct _queue *link;
};

queue *create_queue_node(void)
{
    3   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 ?
        // 재귀호출
    2   enqueue_data(&(*head)->link, data);
}
```

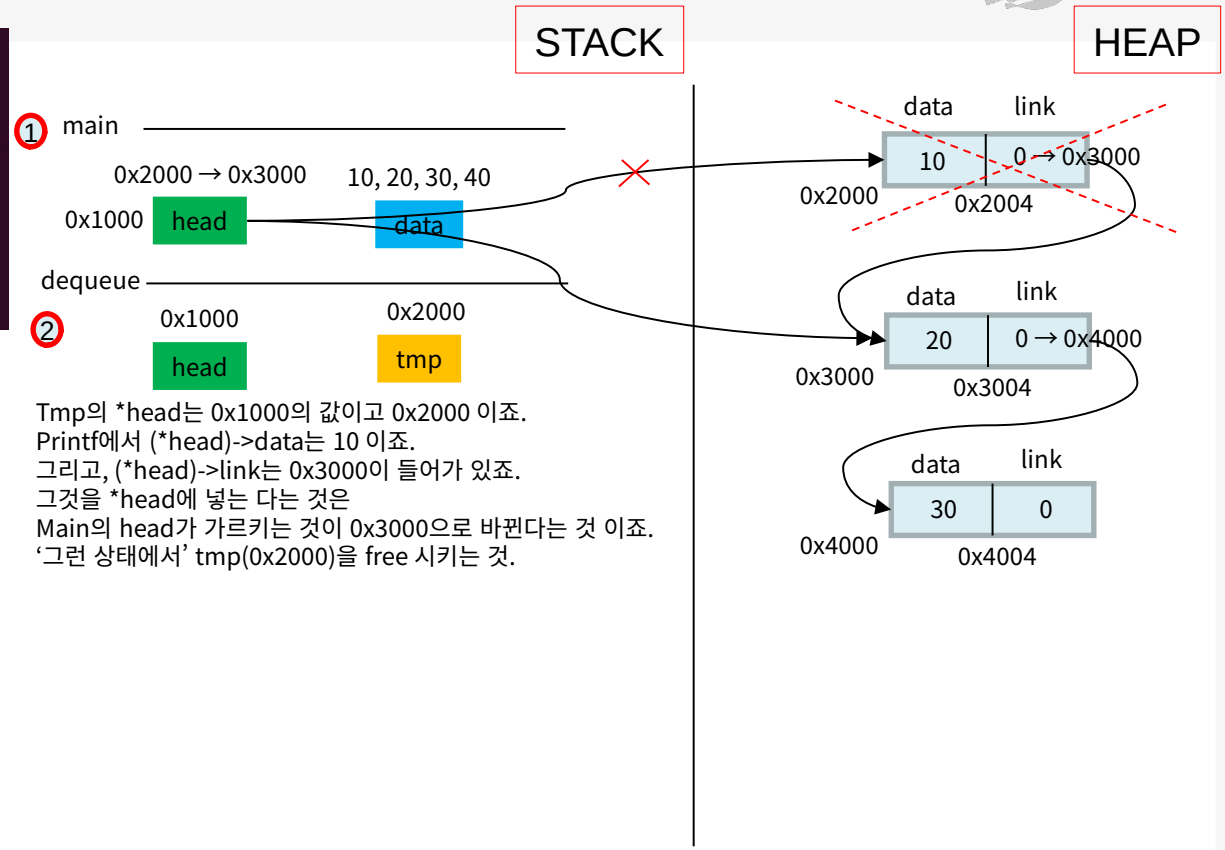
```c
int main(void)
{
        int i;
        queue *head = NULL;
        int data[] = { 10, 20, 30, 40 };

        for (i = 0; i < 4; i++)
        {
    1           enqueue_data(&head, data[i]);
        }
                print_queue(head);
```



Queue 의 dequeue (1)

```
void dequeue_data(queue **head)
{
        if(*head)
        {
            ② queue* tmp = *head;
               printf("dequeue data = %d\n", (*head)->data);
               *head = (*head)->link;
               free(tmp);
        }
        else
        {
               printf("Queue is already empty!\n");
        }
}
```

```
int main(void)
{
    ① int i;
       queue *head = NULL;
       int data[] = { 10, 20, 30, 40 };

       for (i = 0; i < 4; i++)
       {
              enqueue_data(&head, data[i]);
       }
              print_queue(head);

// #if 0 는 주석 넣는 방법 중 하나

       for (i = 0; i < 5; i++)
       {
           ② dequeue_data(&head);
       }

       for (i = 0; i < 4; i++)
       {
              enqueue_data(&head, data[i]);
       }

       // 특정 인덱스에 값 넣기
       enqueue_data_idx(&head, 33, 2);
```

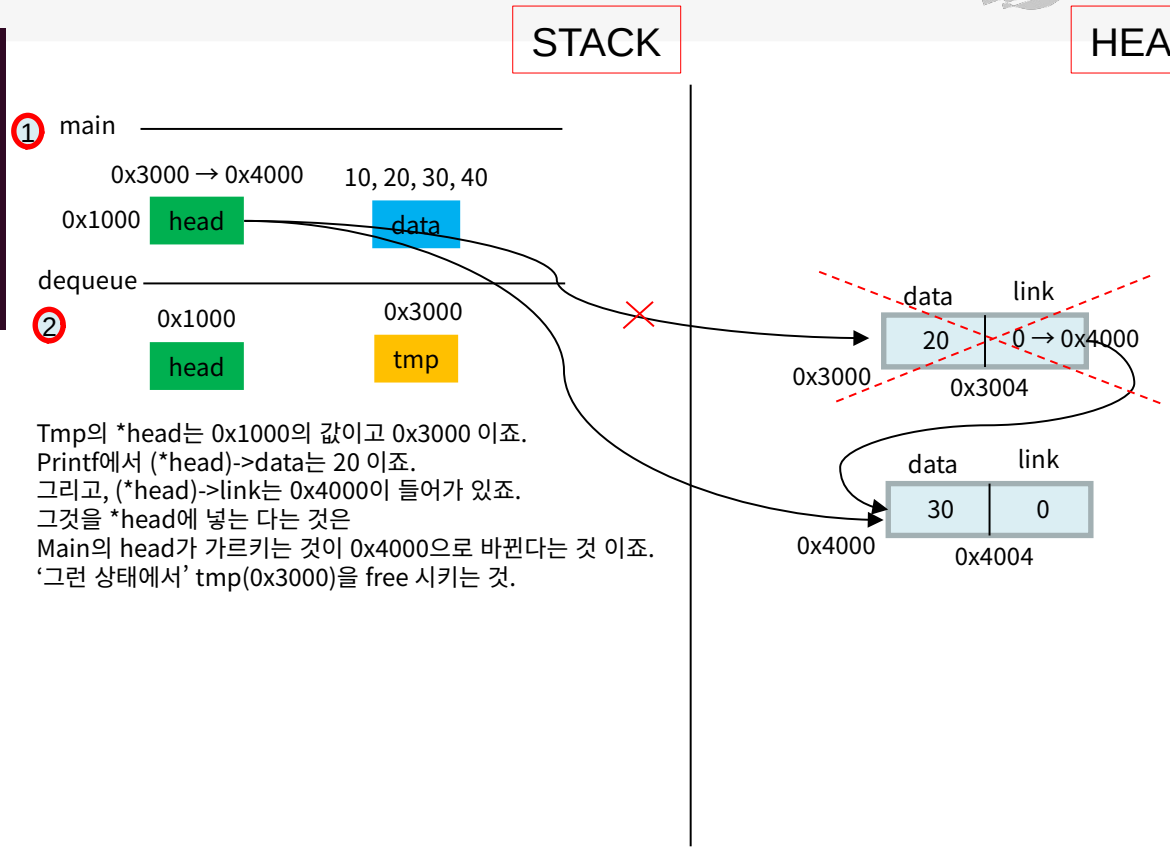


Tmp의 *head는 0x1000의 값이고 0x2000 이죠.
Printf에서 (*head)->data는 10 이죠.
그리고, (*head)->link는 0x3000이 들어가 있죠.
그것을 *head에 넣는 다는 것은
Main의 head가 가르키는 것이 0x3000으로 바뀐다는 것 이죠.
‘그런 상태에서’ tmp(0x2000)을 free 시키는 것.

Queue 의 dequeue (2)

```c
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 already empty!\n");
        }
}
```

```c
int main(void)
{
        int i;
        queue *head = NULL;
        int data[] = { 10, 20, 30, 40 };

        for (i = 0; i < 4; i++)
        {
                enqueue_data(&head, data[i]);
        }
                print_queue(head);

// #if 0 는 주석 넣는 방법 중 하나

        for (i = 0; i < 5; i++)
        {
                dequeue_data(&head);
        }

        for (i = 0; i < 4; i++)
        {
                enqueue_data(&head, data[i]);
        }

        // 특정 인덱스에 값 넣기
        enqueue_data_idx(&head, 33, 2);
```

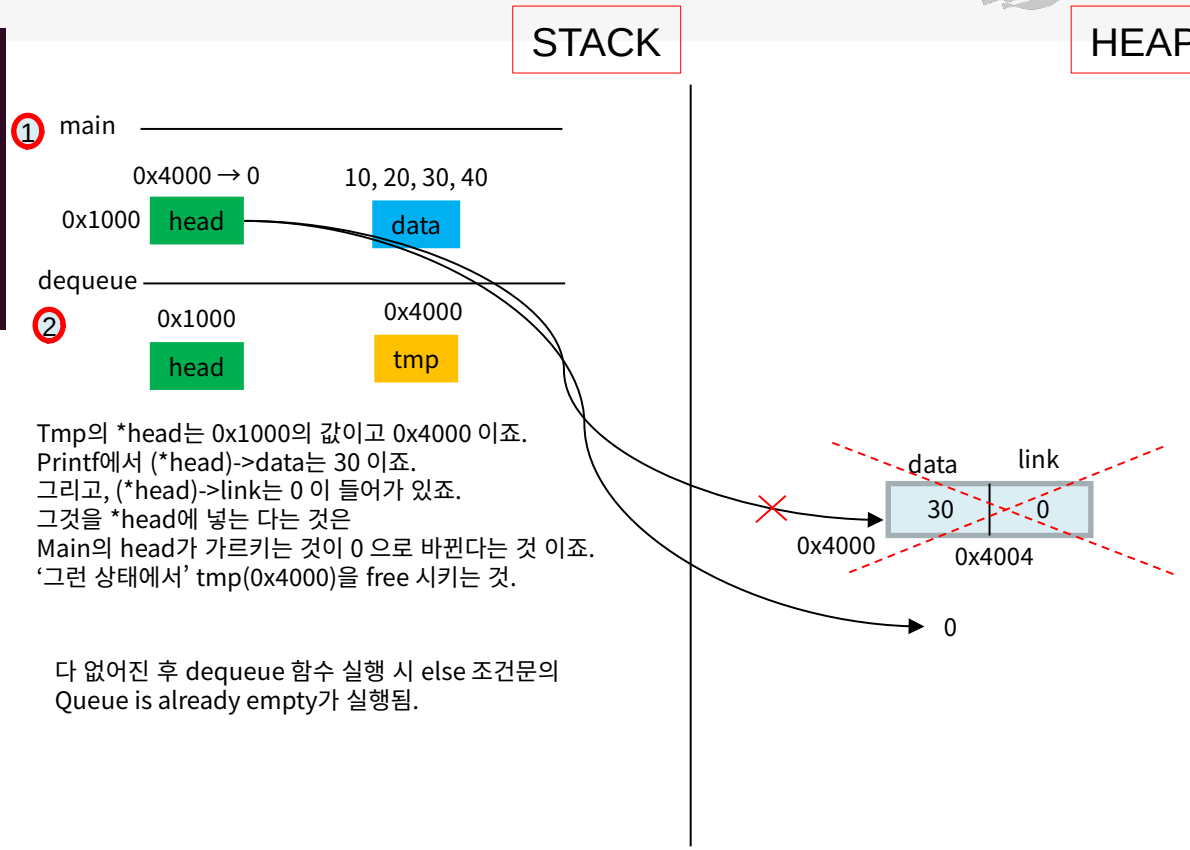


Tmp의 *head는 0x1000의 값이고 0x3000 이죠.
Printf에서 (*head)->data는 20 이죠.
그리고, (*head)->link는 0x4000이 들어가 있죠.
그것을 *head에 넣는 다는 것은
Main의 head가 가르키는 것이 0x4000으로 바뀐다는 것 이죠.
'그런 상태에서' tmp(0x3000)을 free 시키는 것.

Queue 의 dequeue (3)

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

```
int main(void)
{
    1   int i;
        queue *head = NULL;
        int data[] = { 10, 20, 30, 40 };

        for (i = 0; i < 4; i++)
        {
                enqueue_data(&head, data[i]);
        }
                print_queue(head);

// #if 0 는 주석 넣는 방법 중 하나

        for (i = 0; i < 5; i++)
        {
            2   dequeue_data(&head);
        }

        for (i = 0; i < 4; i++)
        {
                enqueue_data(&head, data[i]);
        }

        // 특정 인덱스에 값 넣기
        enqueue_data_idx(&head, 33, 2);
```



Tmp의 *head는 0x1000의 값이고 0x4000 이죠.
Printf에서 (*head)->data는 30 이죠.
그리고, (*head)->link는 0 이 들어가 있죠.
그것을 *head에 넣는 다는 것은
Main의 head가 가르키는 것이 0 으로 바뀐다는 것 이죠.
'그런 상태에서' tmp(0x4000)을 free 시키는 것.

다 없어진 후 dequeue 함수 실행 시 else 조건문의
Queue is already empty가 실행됨.

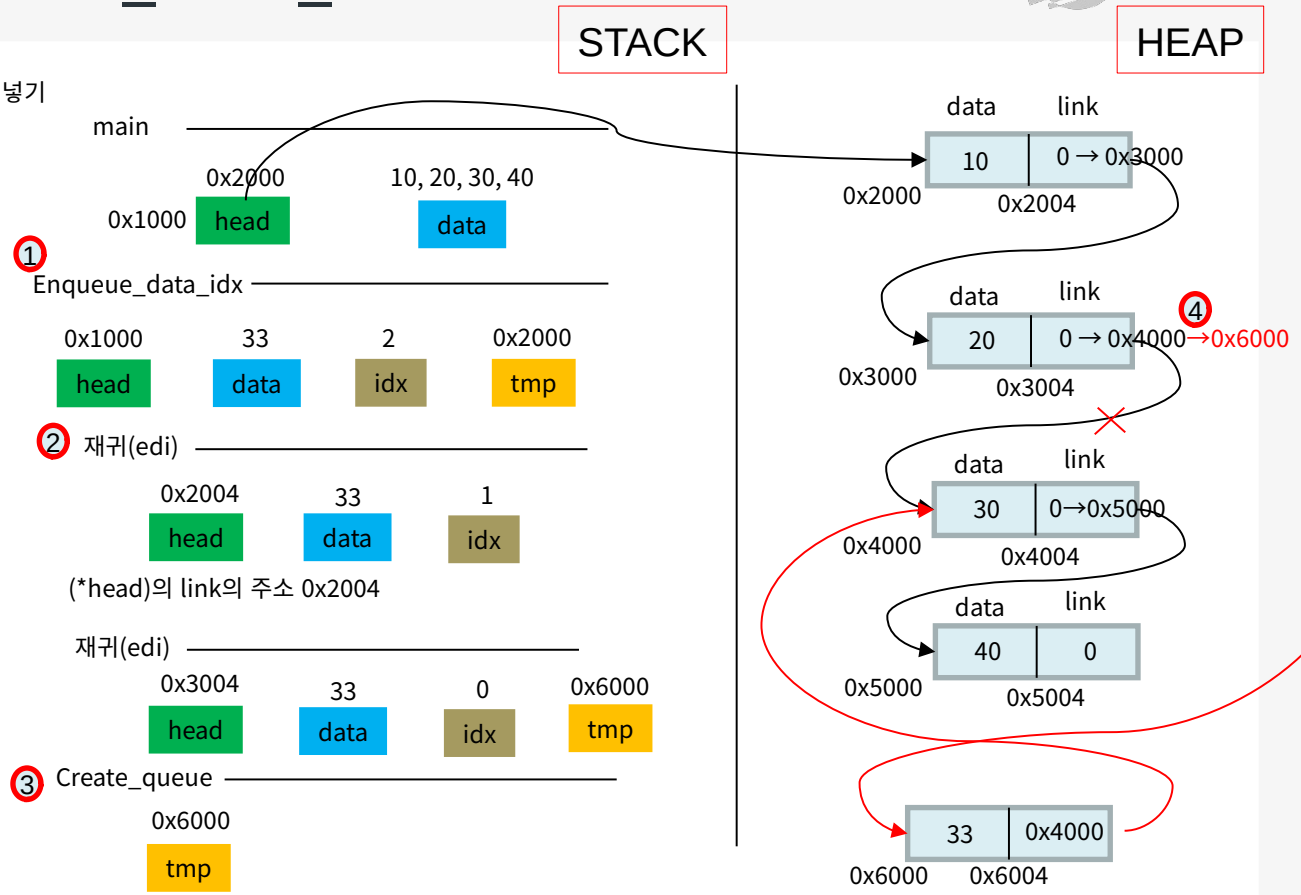
Queue 의 enqueue_data_idx

◆ Queue 구조에서 특정 위치(index) 에 값 집어 넣기

```
① // 특정 인덱스에 값 넣기
  enqueue_data_idx(&head, 33, 2);
```

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

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

        return tmp;
}
```

```
void enqueue_data_idx(queue **head, int data, int idx)
{
      // 중간에 넣는 경우
      if(!idx)        // 맨앞은 0 이니까 index가 0이 되서 참이 된다.
      {
③        queue *tmp = create_queue_node();

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

④           *head = tmp;
            return;
      }

      // 예외 핸들링
      if(!(*head))
      {
            printf("작업이 불가 합니다.\n");
            return;

      }
② enqueue_data_idx(&(*head)->link, data, --idx);
}
```

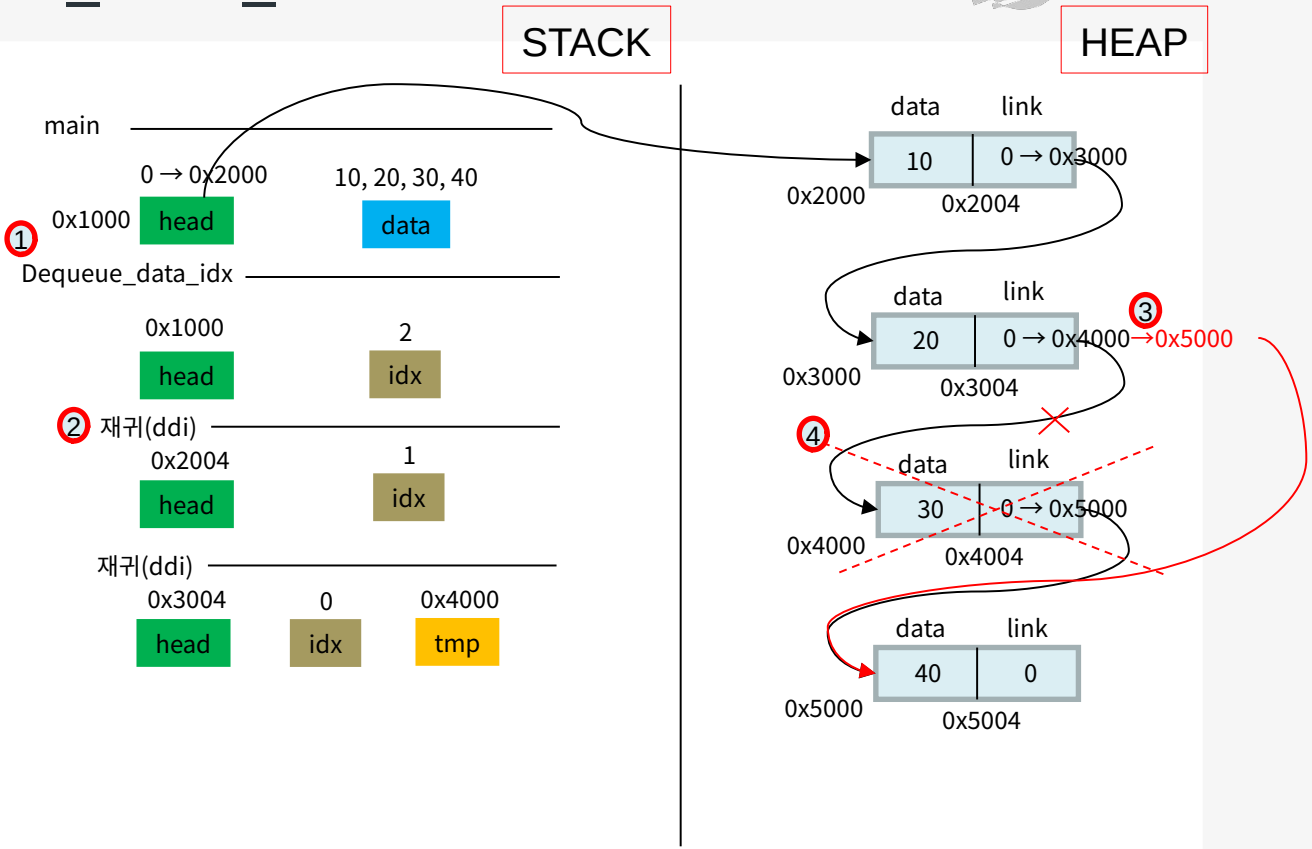


Queue 의 dequeue_data_idx

- Queue 구조에서 특정 위치(index) 에 값 빼기.
  ㄴ index 2 로 예제

```
// 특정 인덱스 값 빼기
dequeue_data_idx(&head, 0);
```

```
void dequeue_data_idx(queue **head, int idx)
{
        if(!idx)        // 맨앞은 0 이니까 index가 0이 되서 참이 된다.
        {
                queue *tmp = *head;

                //tmp->data = data;
        3       (*head) = (*head)->link;

        4       free(tmp);


                return;
        }

        // 예외 핸들링
        if(!(*head))
        {
                printf("작업이 불가 합니다.\n");
                return;

        }

    2   dequeue_data_idx(&(*head)->link, --idx);
}
```



Queue 의 nr_enqueue_data_idx

◆ 재귀호출 없이 특정 index에 enqueue, **nr_enqueue_data_idx(&head, 33, 2)**

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

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

        return tmp;
}
```

```
// 재귀호출 없이 enqueue
// 함수를 호출 할 때 마다의 스택프레임 생성 절차가 사라져서 속도가 더 빠를 수 있다.

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

        //숏컷으로 예외처리 대신함
        //앞에서 미리 판단 한다는 의미임. && 연산자는 모두가 참 이여야 참 이므로
        //loop가 참일 지라도 index가 거짓이면 거짓
        while(loop && idx)
        {
                idx--;
                backup = loop;
                loop = loop->link;
        }

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

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


                // 제일 앞 일 경우 back이 null 이므로 이러한 if문 구성
                if(!backup)
                {
                        *head = tmp;
                }
                else
                {
                        backup->link = tmp;
                }
                return;
        }

        printf("처리가 불가능한 작업 입니다!\n");
}
```