EDDI
Electronic Design
Development Institute

에디로봇아카데미
임베디드 마스터 Lv2 과정

제 1기

2021. 10. 15

김태훈

CONTENTS

EDDI
Electronic Design Development Institute

Double Pointer란?
- 포인터의 포인터. 포인터가 변수의 주소값을 저장하는 변수이므로,
Double Pointer는 포인터의 주소값을 저장함

Double Pointer는 왜 쓸까?
- 현재 call 한 function의 stack frame이 아니라 다른 stack frame에 있는 'pointer'를 제어하고 싶을 때 사용. Parameter로 제어하고 싶은 pointer의 주소값을 받으면 현재 함수의 stack frame에서 다른 stack frame으로 접근이 가능하다.

Single Pointer를 쓰면 안 되나?
- C는 parameter 전달 방식이 Call by value이기 때문에 single pointer를 전달한 다음에 수정을 해도 main의 stack frame의 변수는 변경되지 않는다. 따라서 return 과정을 통해서 main함수에서 변수를 변경해야 하는 과정이 추가로 필요하다.

CONTENTS

EDDI
Electronic Design
Development Institute

Double Pointer를 이용하면 return을 이용하지 않아도 다른 stack frame에 있는 값을 변경할 수 있다.
물론 pointer만 받아도 다른 stack frame에 있는 variable의 값을 변경할 수 있지만 Pointer의 값(주소값)을 바꾸려면 double pointer를 써야 한다.

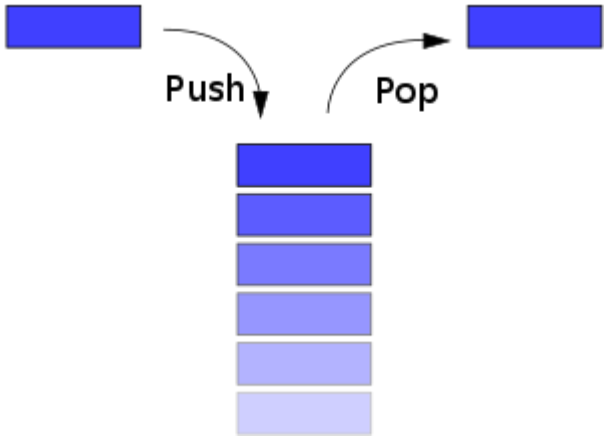
이번 수업 시간에 double pointer을 쓴 이유는 정리하면
1) Function의 stack frame에서
2) Main stack frame에 있는 pointer에 있는 값을
3) Heap에 할당되어있는 struct의 주소로 수정하기 위해서이다.

CONTENTS

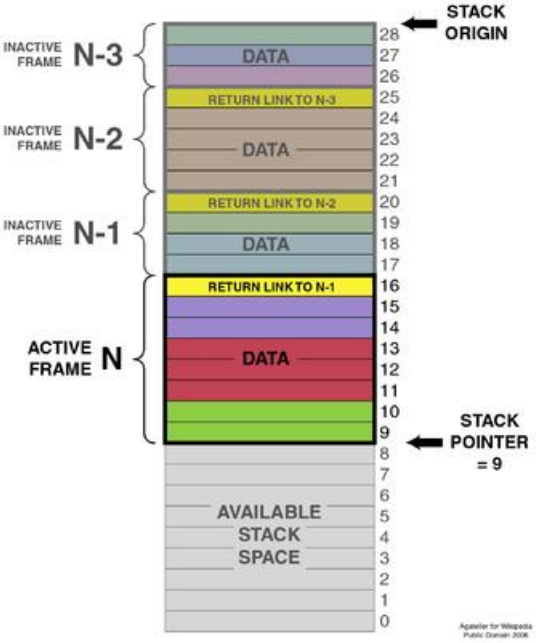
EDDI
Electronic Design Development Institute

Stack이란?

LIFO : 후입선출 구조를 가지는 자료구조.



Abstract stack



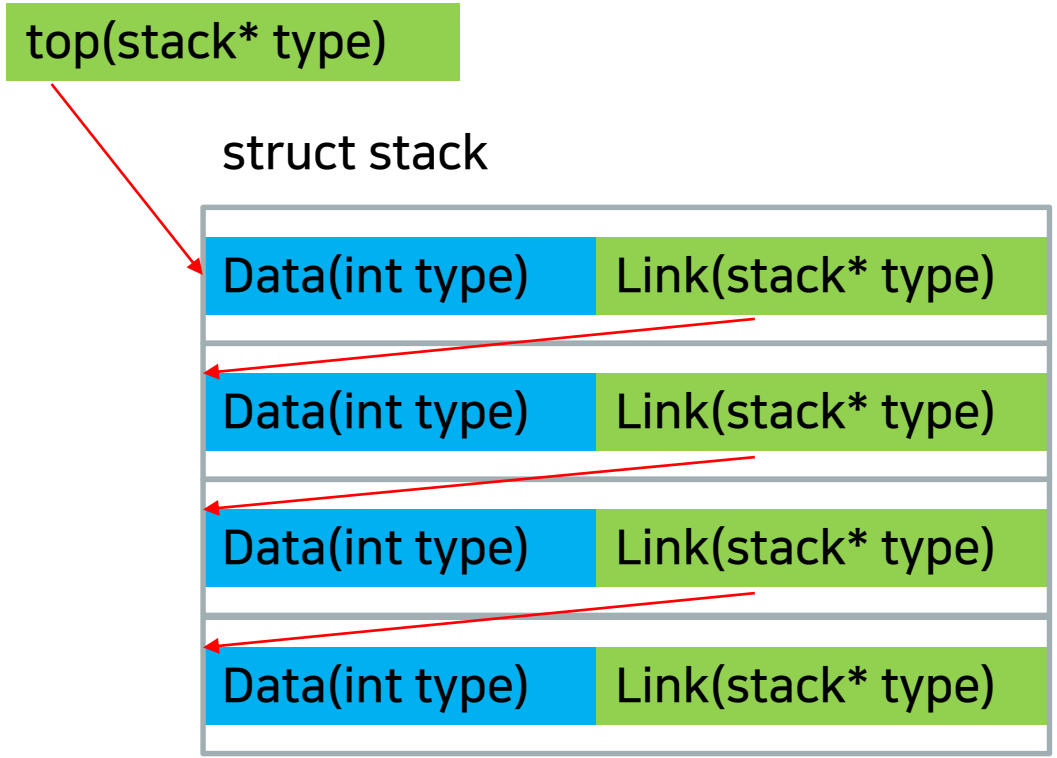
Memory using stack

CONTENTS

EDDI
Electronic Design Development Institute

Stack 구조체

```
typedef struct _stack stack;
struct _stack
{
    int data;
    struct _stack *link;
};
```

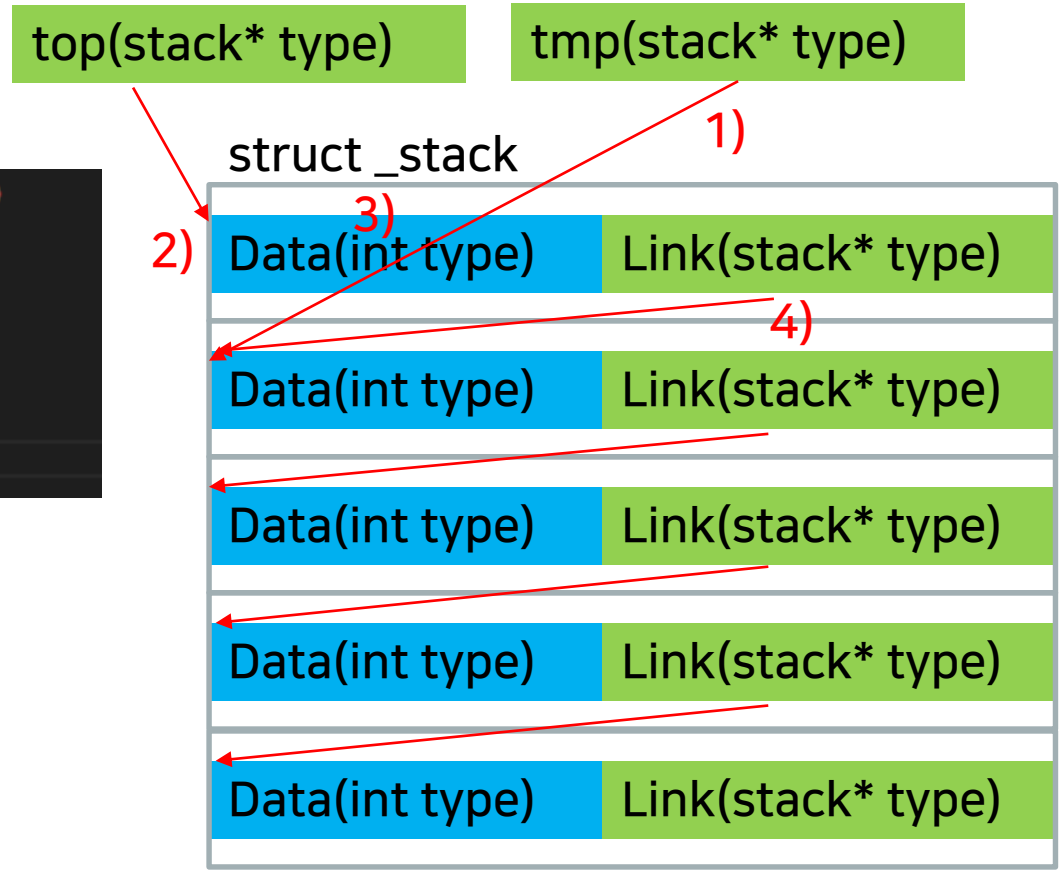


CONTENTS

Push

```
void push_data(stack **top, int data)
{
1) stack *tmp = *top;

2) *top = create_stack_node();
3) (*top)->data = data;
4) (*top)->link = tmp;
}
```



CONTENTS

EDDI
Electronic Design Development Institute

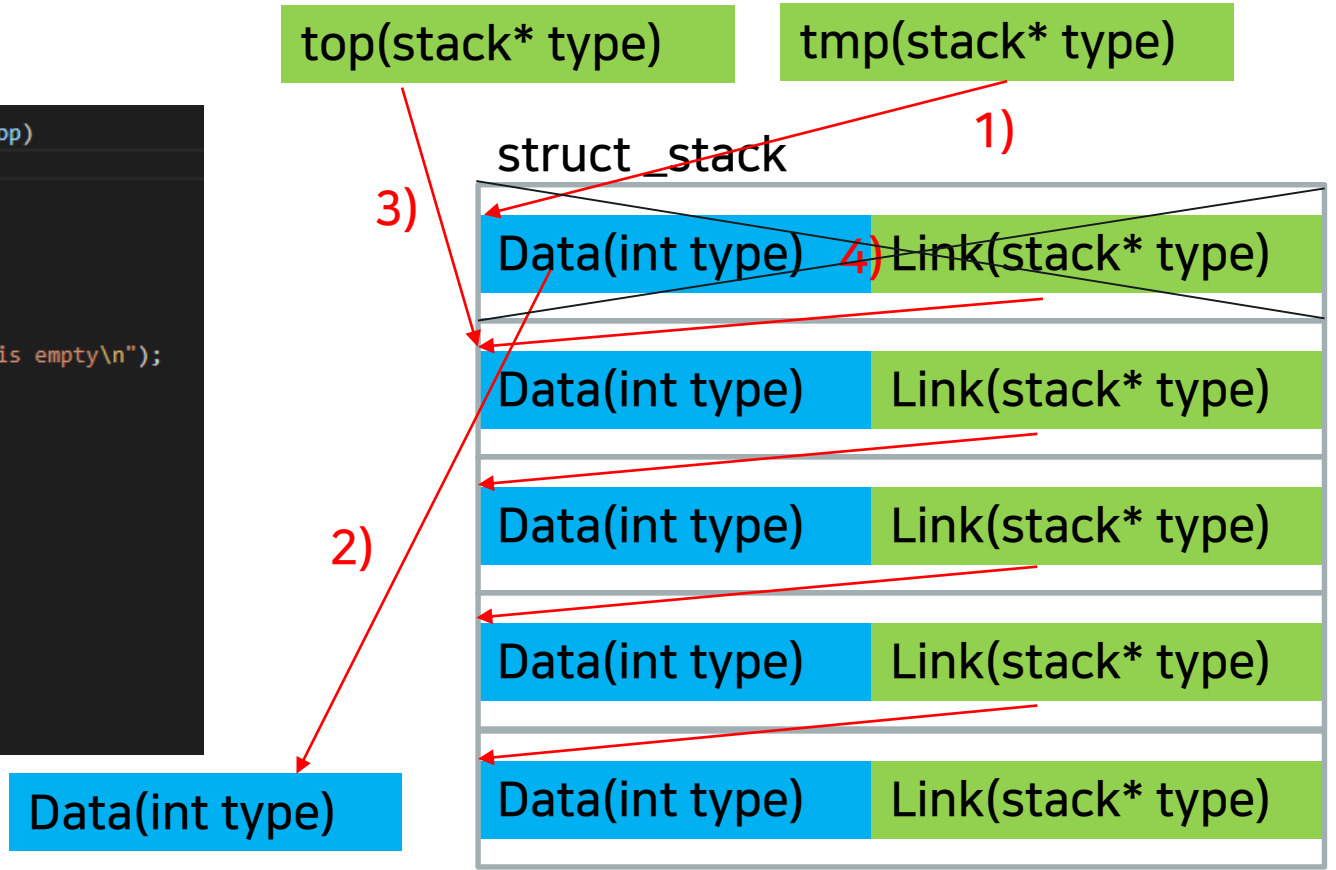
Pop

```
int pop_data(stack **top)
{
    int data;
    stack *tmp;

    if (!(*top))
    {
        printf("Stack is empty\n");
        return -1;
    }

1)  tmp = *top;
2)  data = tmp->data;
3)  *top = tmp->link;
4)  free(tmp);

    return data;
}
```



CONTENTS

EDDI
Electronic Design Development Institute

main

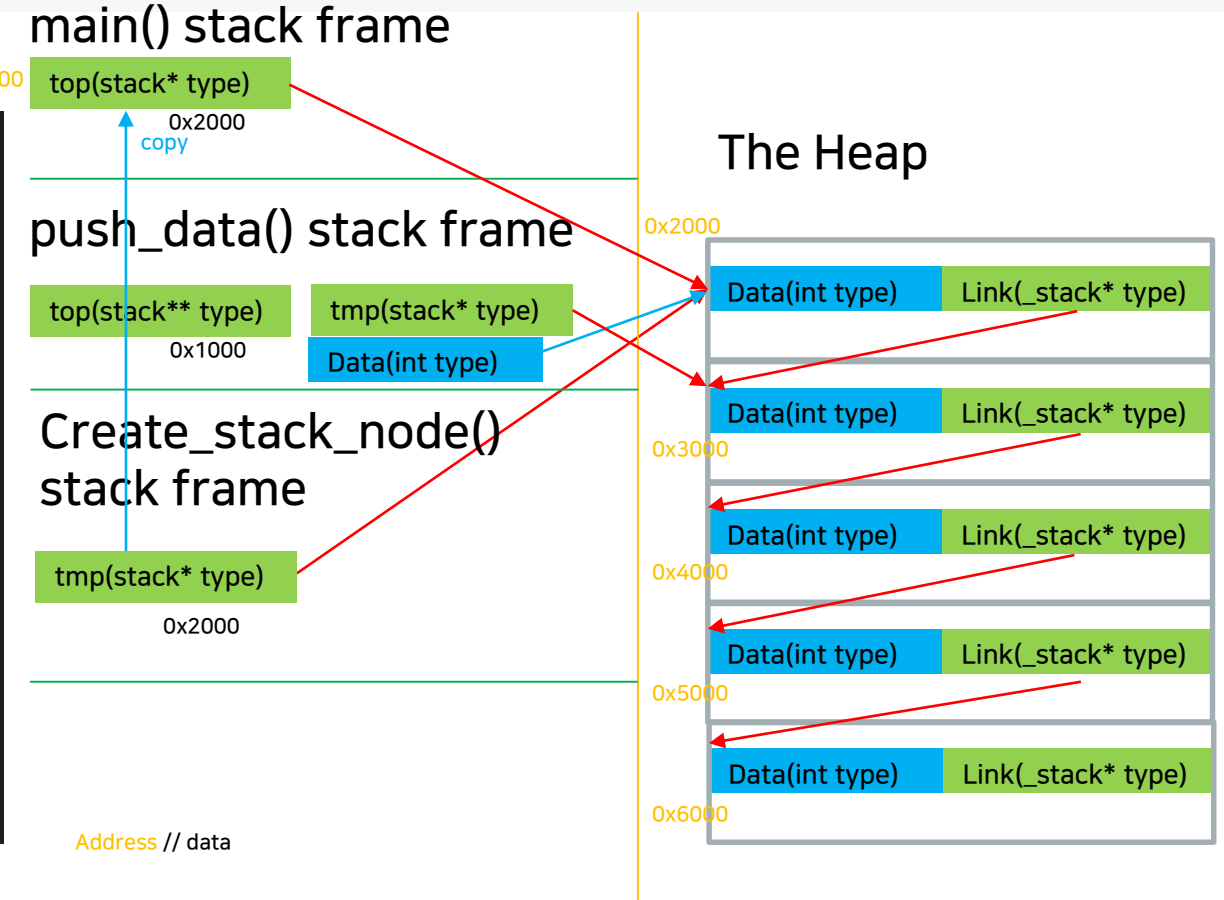
```
int main(void)
{
    stack *top = NULL;
    int data[] = { 10, 20, 30, 40, 50, 60, 70 };
    int i;

    for (i = 0; i < 7; i++)
    {
        push_data(&top, data[i]);
    }

    print_stack_data(top);

    for (i = 0; i < 8; i++)
    {
        printf("pop = %d\n", pop_data(&top));
        print_stack_data(top);
    }

    return 0;
}
```

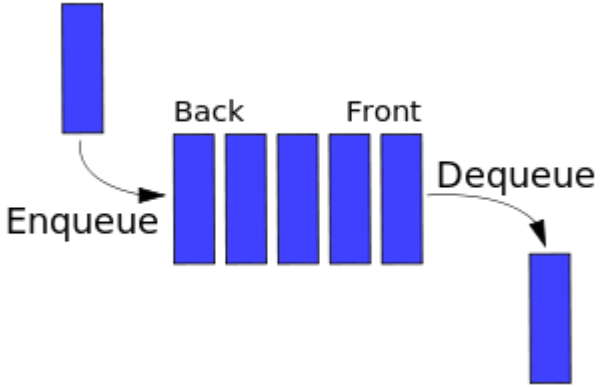


CONTENTS

EDDI
Electronic Design
Development Institute

QUEUE이란?

FIFO : 선입선출 구조를 가지는 자료구조.



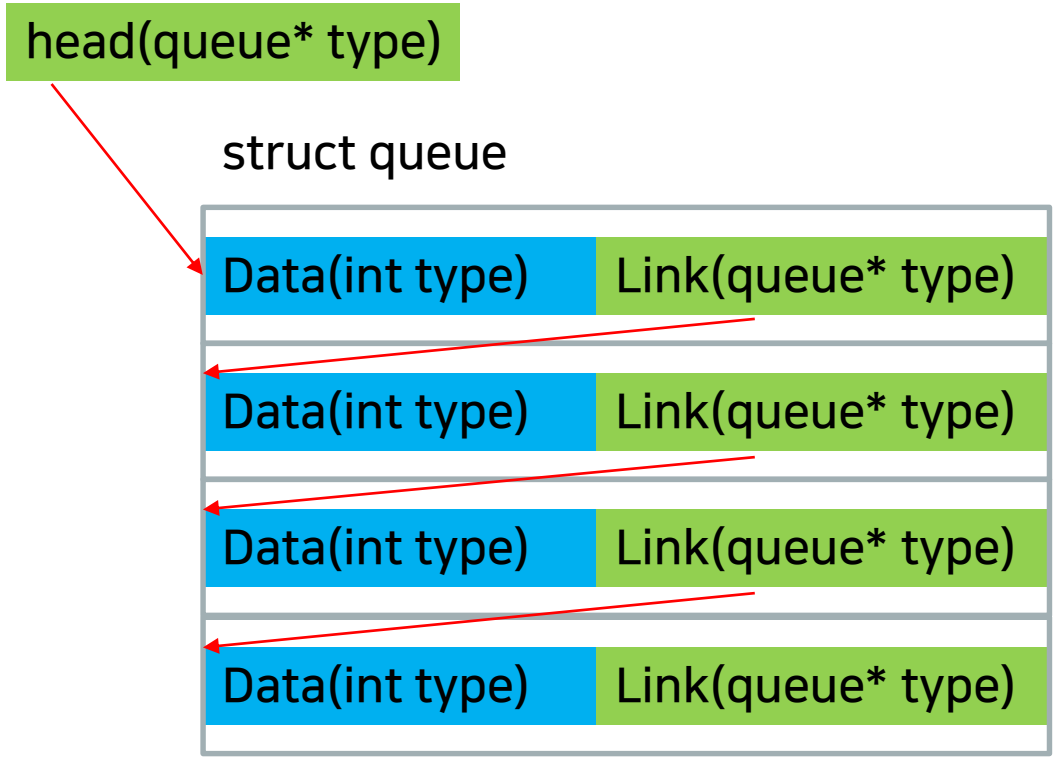
Abstract queue

CONTENTS

EDDI
Electronic Design Development Institute

queue 구조체

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



Stack 구조체랑 비슷하게 생김

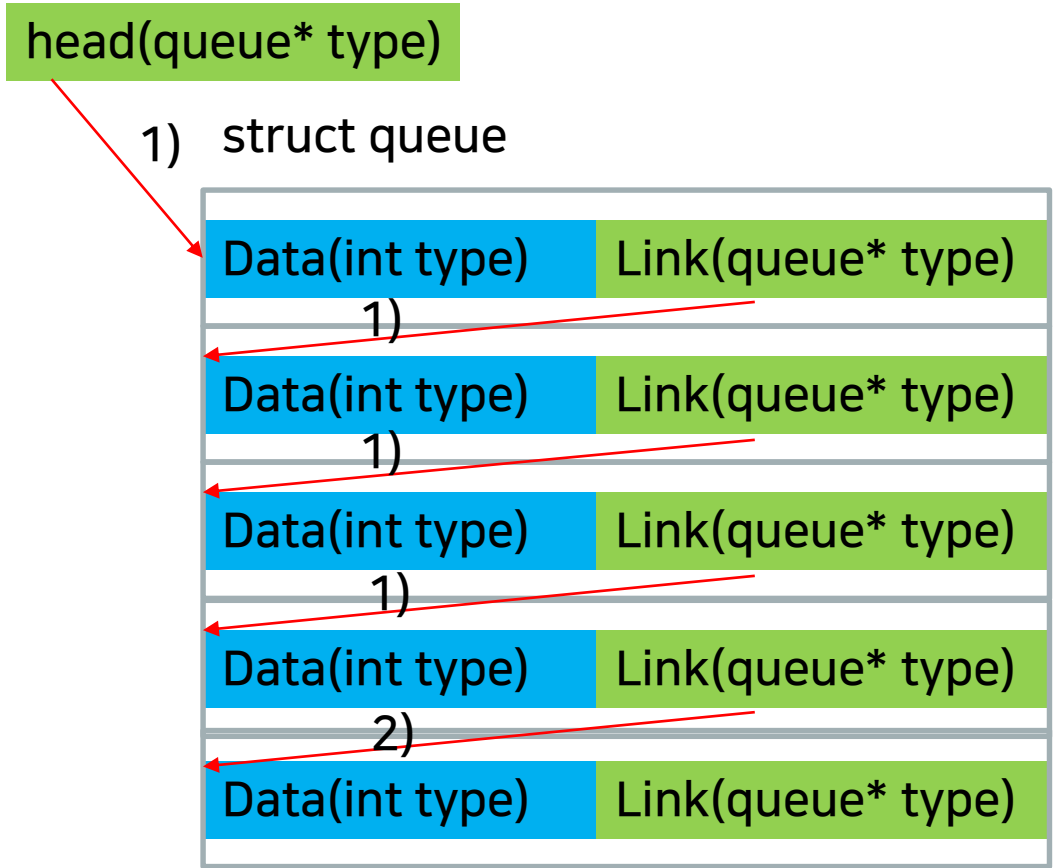
CONTENTS

EDDI
Electronic Design Development Institute

enqueue

```
void enqueue_data(queue **head, int data)
{
    if(!(*head))
    {
        *head = create_queue_node();
        (*head)->data = data;
        return;
    }
    enqueue_data(&(*head)->link,data);
    // TODO
    //
}
#endif
```

1) Recursion을 통해 link가 NULL일 때까지 내려감
2) 마지막 link에 node를 생성하고 data를 넣음



CONTENTS

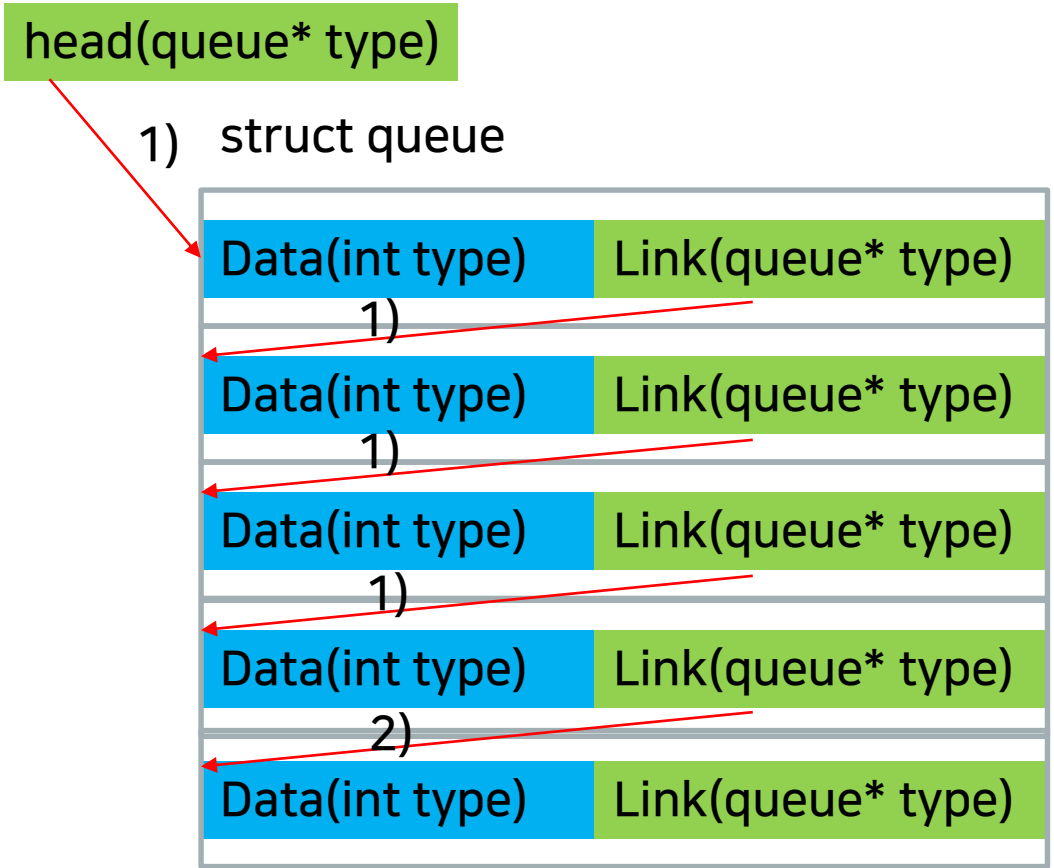
EDDI
Electronic Design Development Institute

enqueue(not recursion)

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

    while(tmp->link)
    {
        tmp=tmp->link;
    }
    tmp->link = create_queue_node();
    (tmp->link)->data = data;
}
```

1) While문을 이용해 link가 null일때까지 내려감
2) 마지막 link에 node를 생성하고 data를 넣음

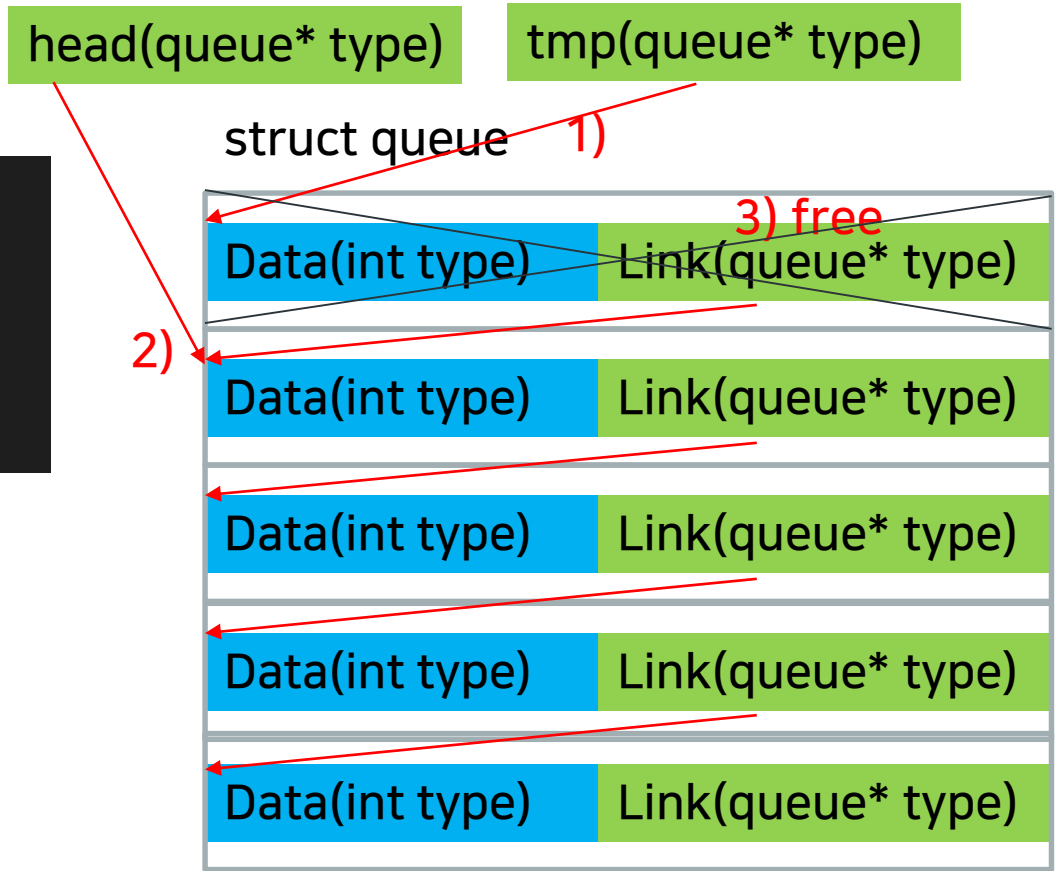


CONTENTS

EDDI
Electronic Design Development Institute

dequeue

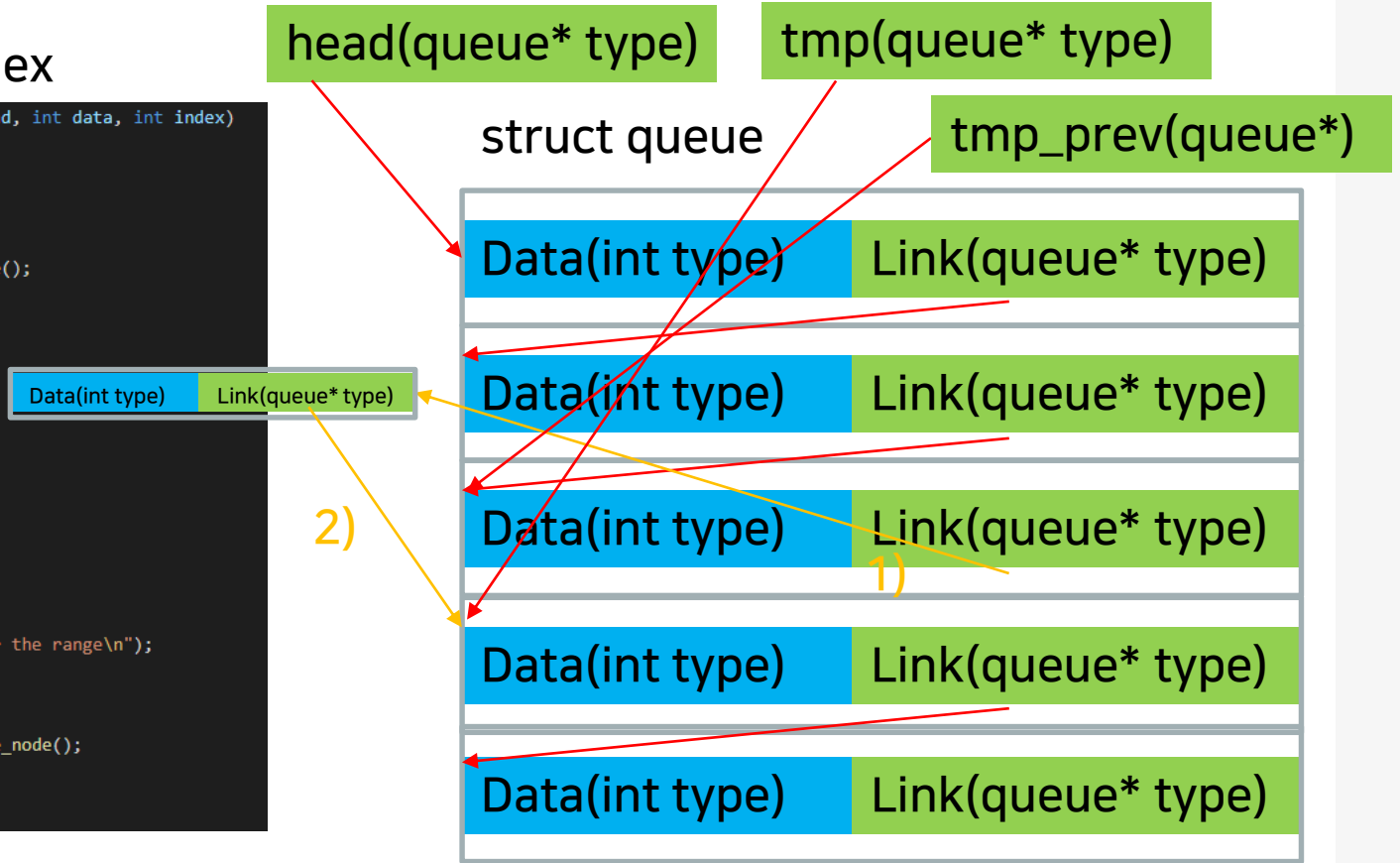
```
void dequeue_data(queue** head)
{
    if(*head)
    {
    1) queue* tmp = *head;
    2) *head=(*head)->link;
    3) free(tmp);
    }
    else
        printf("Cannot dequeue - Already Empty queue\n");
}
```



CONTENTS

enqueue at index

```
void enqueue_data_idx(queue** head, int data, int index)
{
    queue* tmp = *head;
    queue* tmp_prev = NULL;
    if(!index)
    {
        *head = create_queue_node();
        (*head)->data = data;
        (*head)->link = tmp;
        return;
    }
    while(index)
    {
        if(tmp)
        {
            tmp_prev = tmp;
            tmp=tmp->link;
            index--;
        }
        else
        {
            printf("index is over the range\n");
            return;
        }
    }
 1) tmp_prev->link = create_queue_node();
    tmp_prev->link->data = data;
 2) tmp_prev->link->link = tmp;
}
```



CONTENTS

EDDI
Electronic Design Development Institute

dequeue at index

```
void dequeue_data_idx(queue** head, int index)
{
    queue* tmp = *head;
    queue* tmp_prev = NULL;
    if(!index)
    {
        *head = (*head)->link;
        free(tmp);
        return;
    }
    while(index)
    {
        if(tmp)
        {
            tmp_prev=tmp;
            tmp=tmp->link;
            index--;
        }
        else
        {
            printf("index is over the range\n");
            return;
        }
    }
  1) tmp_prev->link = tmp_prev->link->link;
  2) free(tmp);
}
```