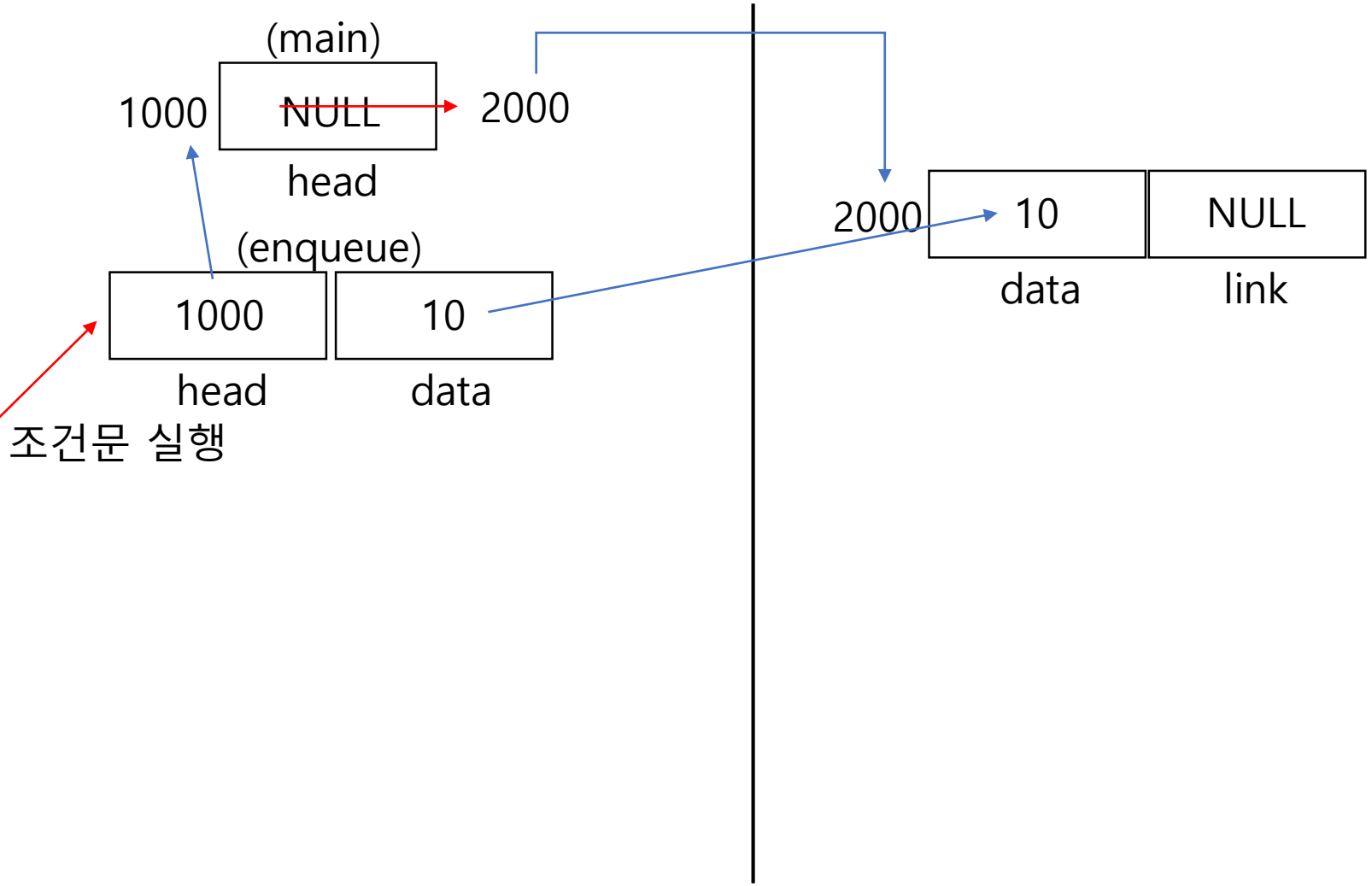
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 문한나
mhn97@naver.com

연결리스트 예제 그림 그리기 –queue

```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```

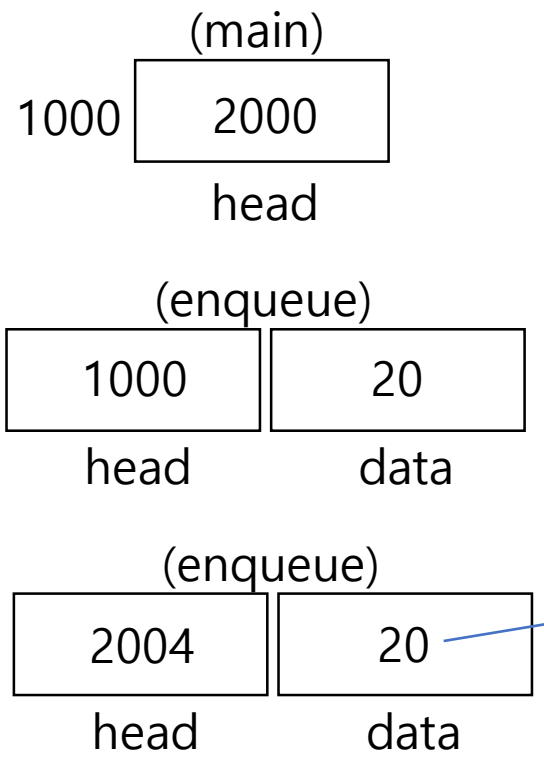
```c
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <time.h>

typedef struct __queue{
        int data;
        struct __queue *link;
}queue;

queue *get_node(){
        queue *tmp;
        tmp = (queue *)malloc(sizeof(queue));
        tmp -> link = NULL;
        return tmp;
}

void enqueue(queue **head,int data){
        if(*head == NULL){
                *head = get_node();
                (*head)->data=data;
                return;
        }
        enqueue(&(*head)->link,data);

}
```

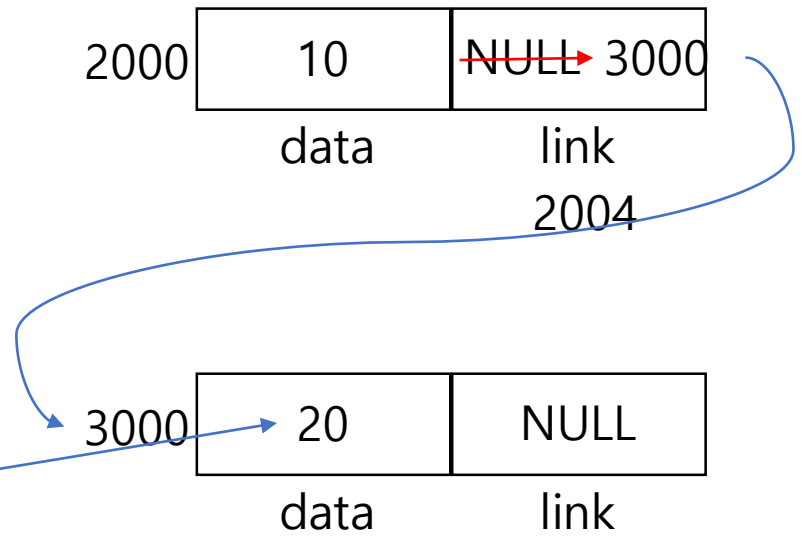


```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```

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

typedef struct __queue{
        int data;
        struct __queue *link;
}queue;

queue *get_node(){
        queue *tmp;
        tmp = (queue *)malloc(sizeof(queue));
        tmp -> link = NULL;
        return tmp;
}

void enqueue(queue **head,int data){
        if(*head == NULL){
                *head = get_node();
                (*head)->data=data;
                return;
        }
        enqueue(&(*head)->link,data);
}
```

재귀호출

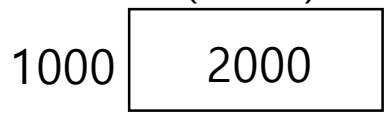




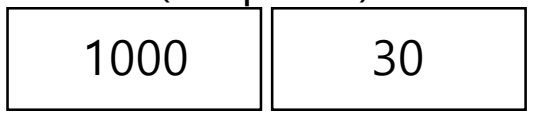
```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```

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

typedef struct __queue{
        int data;
        struct __queue *link;
}queue;

queue *get_node(){
        queue *tmp;
        tmp = (queue *)malloc(sizeof(queue));
        tmp -> link = NULL;
        return tmp;
}

void enqueue(queue **head,int data){
        if(*head == NULL){
                *head = get_node();
                (*head)->data=data;
                return;
        }
        enqueue(&(*head)->link,data);
}
```

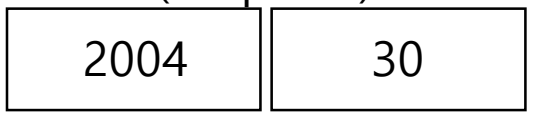
(main)



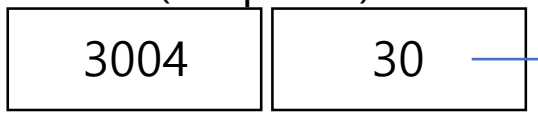
(enqueue)



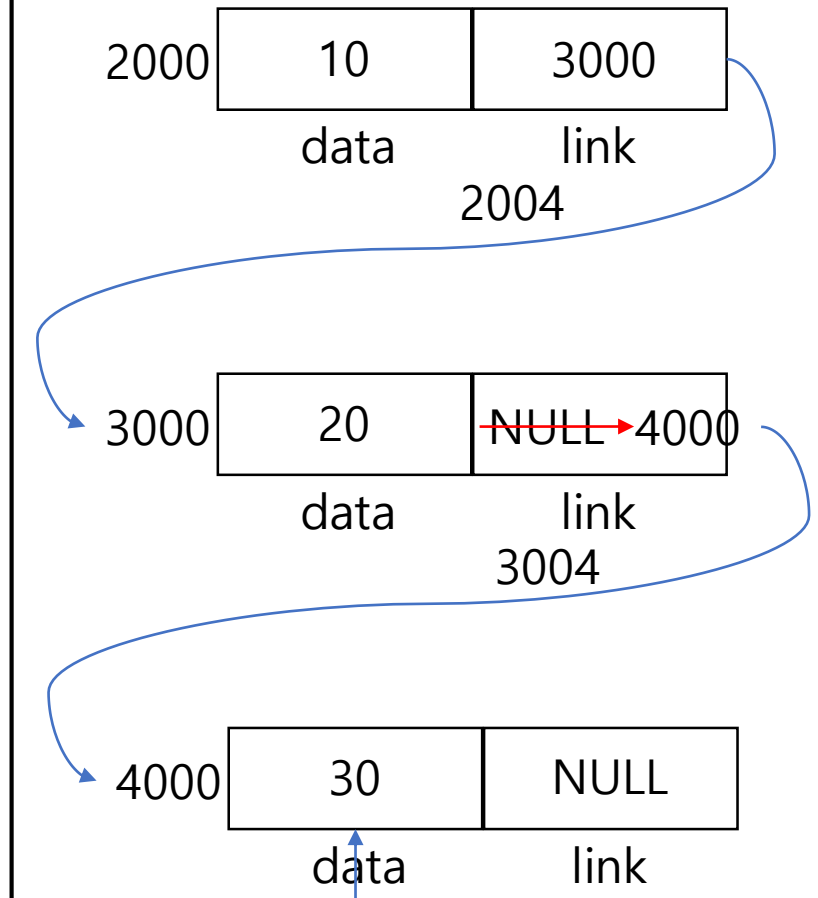
(enqueue)



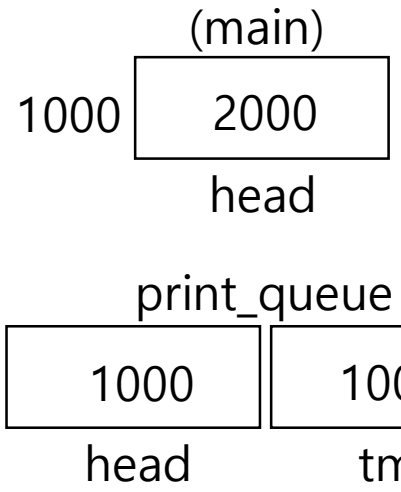
(enqueue)

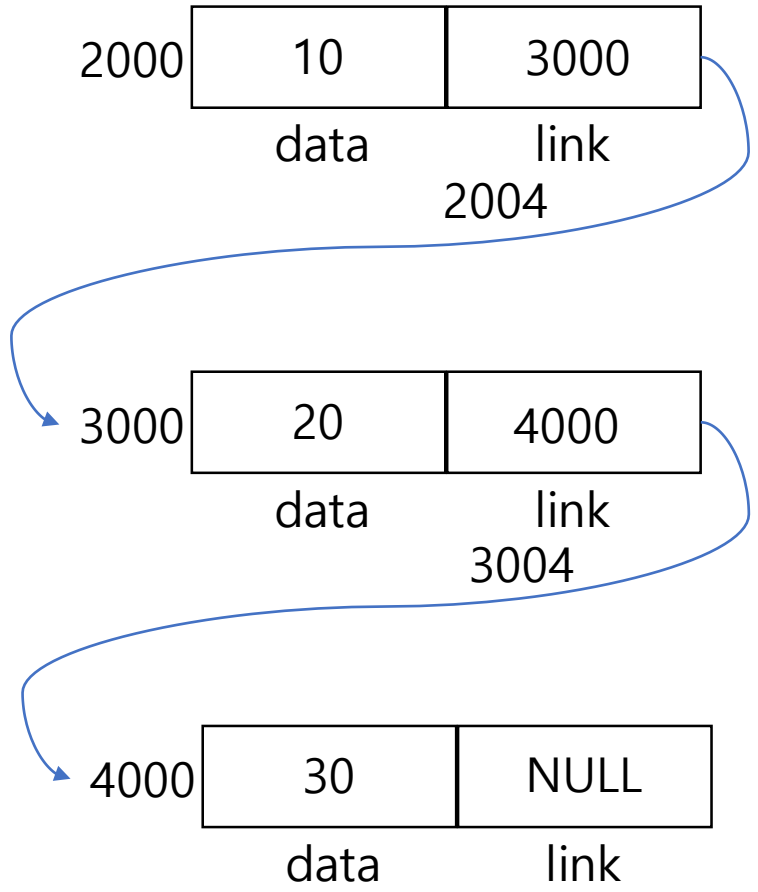


재귀호출



```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```





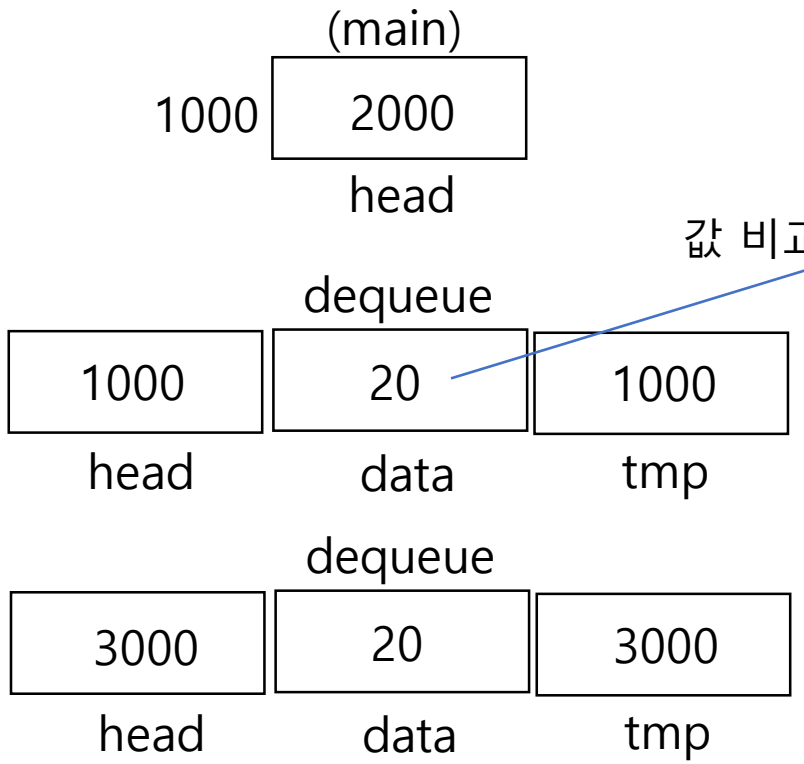

```c
void print_queue(queue *head){
        queue *tmp = head;
        while(tmp){
                printf("%d\n",tmp->data);
                tmp = tmp->link;
        }
}
```

링크를 따라가며 값을 찍는다

```
mhn@mhn-900X3L:~/my_proj/c/8_h$ ./a.out
10
20
30
```

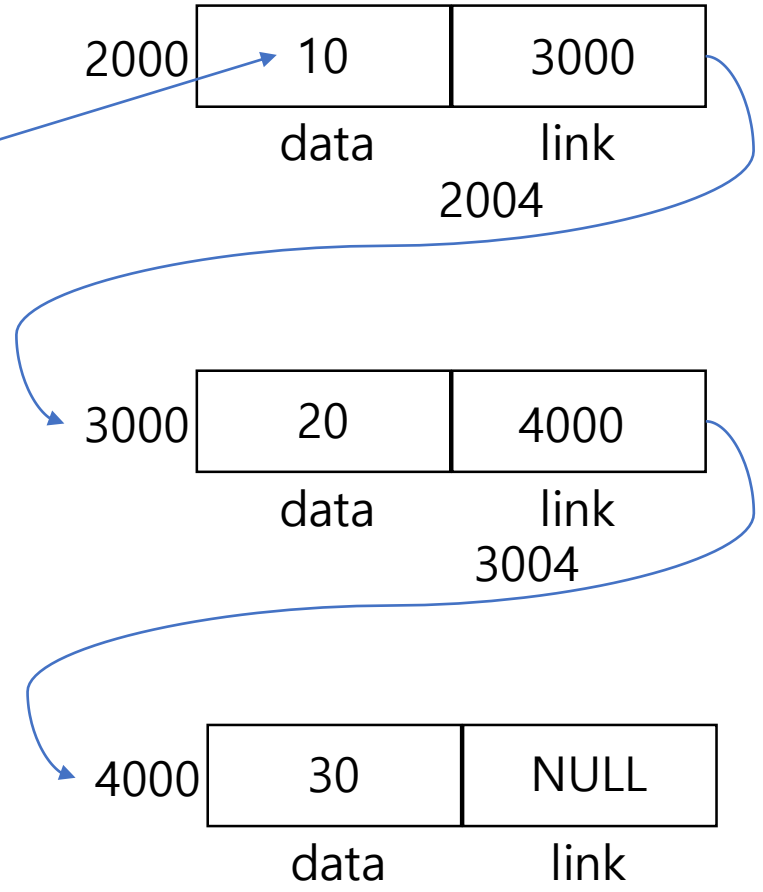
```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```

```c
queue *dequeue(queue *head,int data){
        queue *tmp = head;
        if(tmp == NULL)
                printf("there are no data that you delete\n");

        if(head -> data != data)
                head->link = dequeue(head->link,data);
        else{
                //queue *res = head->link;
                printf("Now you delete %d\n",data);
                free(tmp);
                return head->link;
        }
        return head;
}
```

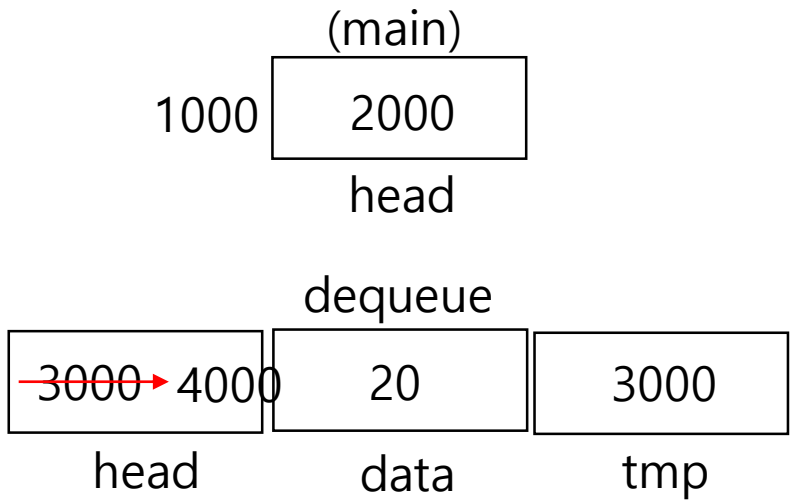


값 비교

heap에 들어있는 data와
값이 다르면 head의 link값을 가지고
재귀호출을 한다



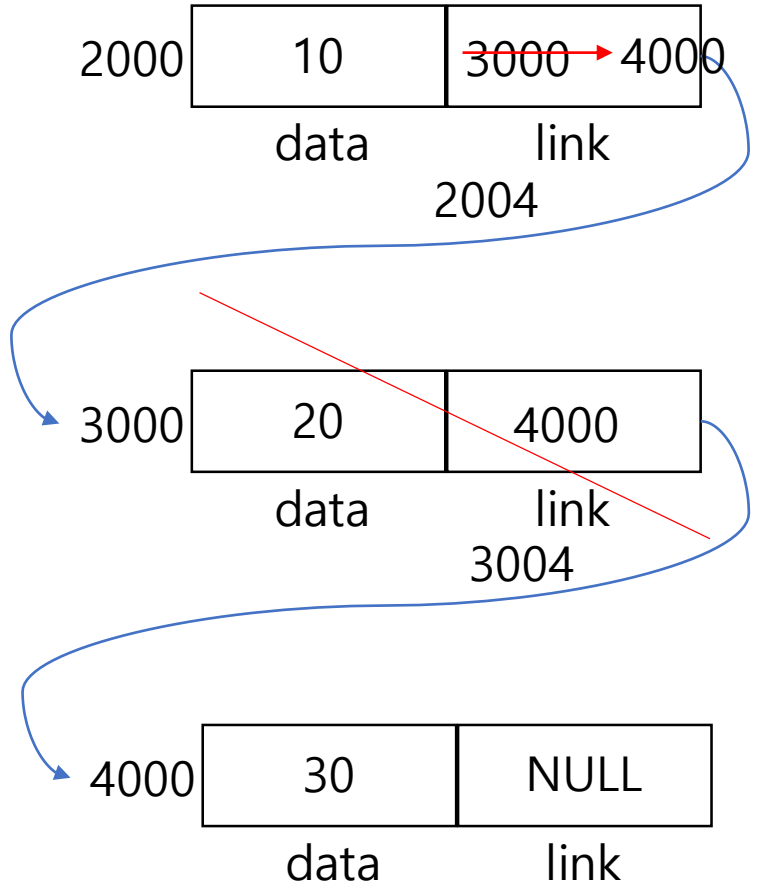
```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```



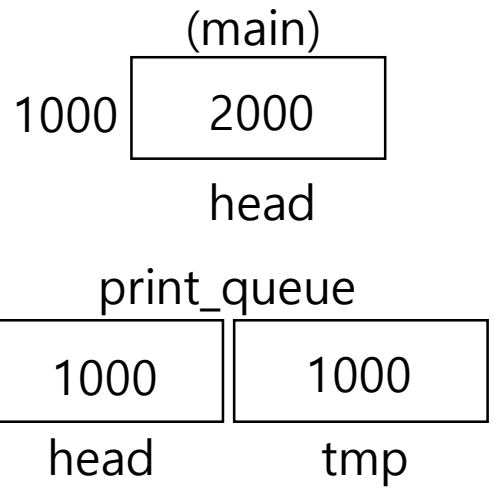

```c
queue *dequeue(queue *head,int data){
        queue *tmp = head;
        if(tmp == NULL)
                printf("there are no data that you delete\n");

        if(head -> data != data)
                head->link = dequeue(head->link,data);
        else{
                //queue *res = head->link;
                printf("Now you delete %d\n",data);
                free(tmp);
                return head->link;
        }
        return head;
}
```

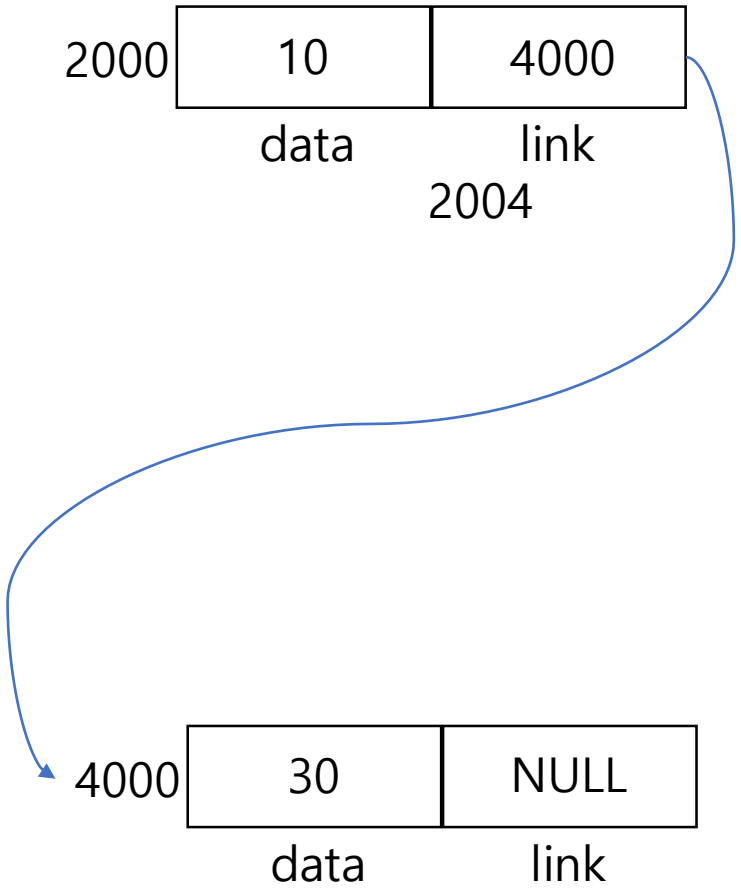
heap에 들어있는 data와
값이 같으면 데이터 할당을 해제한다

head는 해제한 링크의 주소를 따라간다



```c
int main(void){

        int i;
        queue *head = NULL;

        srand(time(NULL));
        for(i=0;i<3;i++)
        enqueue(&head,(i+1)*10);
        print_queue(head);

        head = dequeue(head,20);
        print_queue(head);

        return 0;

}
```






```c
void print_queue(queue *head){
        queue *tmp = head;
        while(tmp){
                printf("%d\n",tmp->data);
                tmp = tmp->link;
        }
}
```

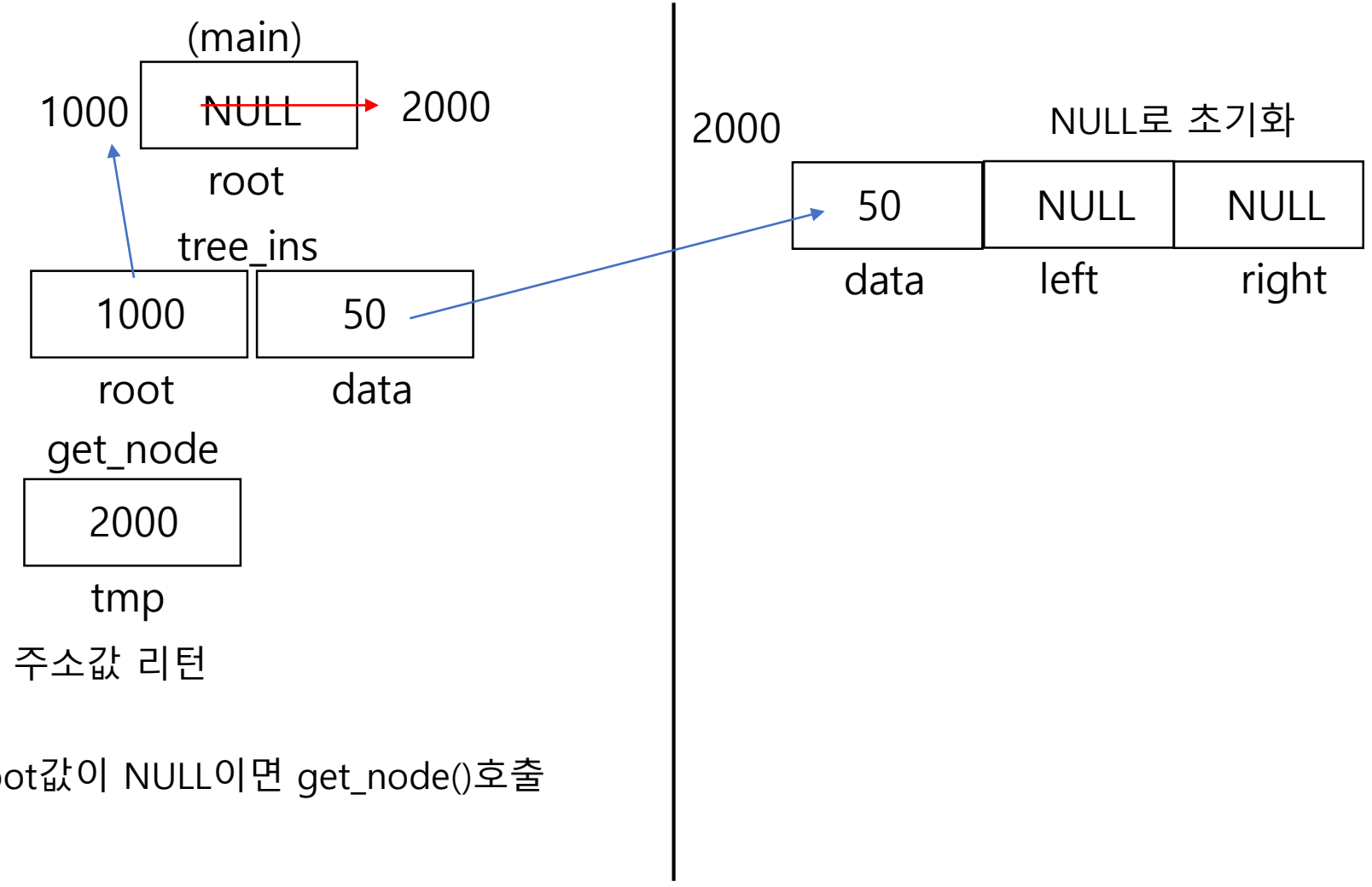
링크를 따라가며 값을 찍는다

```
.mhn@mhn-900X3L:~/my_proj/c/9_s$ ./a.out
10
20
30
Now you delete 20
10
30
mhn@mhn-900X3L:~/my_proj/c/9_s$
```

연결리스트 예제 그림 그리기 –tree

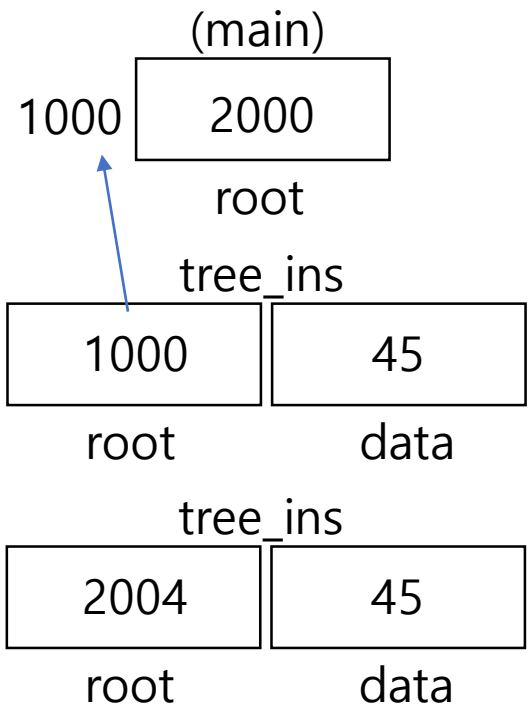
```c
int main(void){

        int i;
        int data[14] = {50,45,73,32,48,46,16,37,120,47,130,127,124};

        tree *root = NULL;

        for(i=0; data[i]; i++)
                tree_ins(&root,data[i]);

        print_tree(root);

        /*delete_tree(root,50);
          printf("after delete\n");

          print_tree(root);*/

        return 0;

}
```

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

struct node{
        int data;
        struct node *left;
        struct node *right;
};
typedef struct node tree;

tree *get_node(){

        tree *tmp;
        tmp = (tree *)malloc(sizeof(tree));
        tmp -> right = NULL;
        tmp -> left = NULL;
        return tmp;

}

void tree_ins(tree **root,int data){

        if(*root == NULL){
                *root = get_node();
                (*root)->data=data;
                return;
        }else if((*root)->data > data)
                tree_ins(&(*root)->left,data);
        else if((*root)->data < data)
                tree_ins(&(*root)->right,data);

}
```



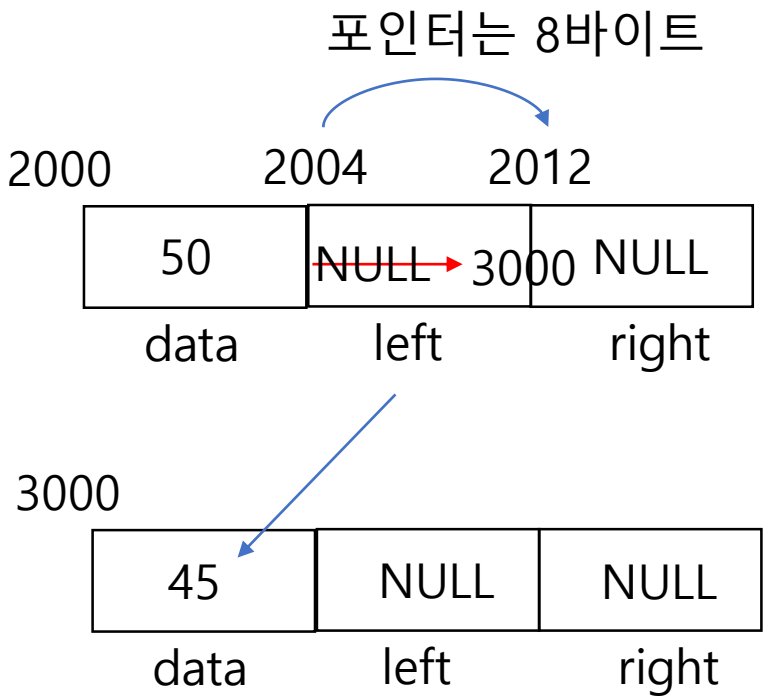
```c
int main(void){

        int i;
        int data[14] = {50,45,73,32,48,46,16,37,120,47,130,127,124};

        tree *root = NULL;

        for(i=0; data[i]; i++)
                tree_ins(&root,data[i]);

        print_tree(root);

        /*delete_tree(root,50);
          printf("after delete\n");

          print_tree(root);*/

        return 0;

}
```

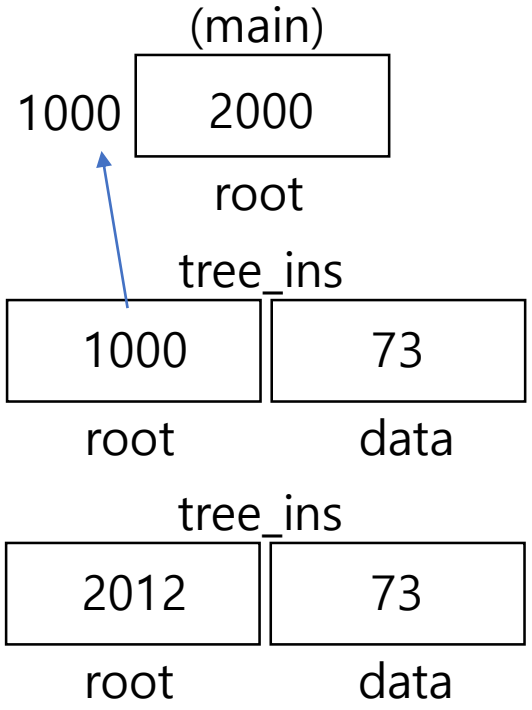
```c
#include <stdio.h>
#include <malloc.h>

struct node{
        int data;
        struct node *left;
        struct node *right;
};
typedef struct node tree;

tree *get_node(){

        tree *tmp;
        tmp = (tree *)malloc(sizeof(tree));
        tmp -> right = NULL;
        tmp -> left = NULL;
        return tmp;

}

void tree_ins(tree **root,int data){

        if(*root == NULL){
                *root = get_node();
                (*root)->data=data;
                return;
        }else if((*root)->data > data)
                tree_ins(&(*root)->left,data);
        else if((*root)->data < data)
                tree_ins(&(*root)->right,data);

}
```

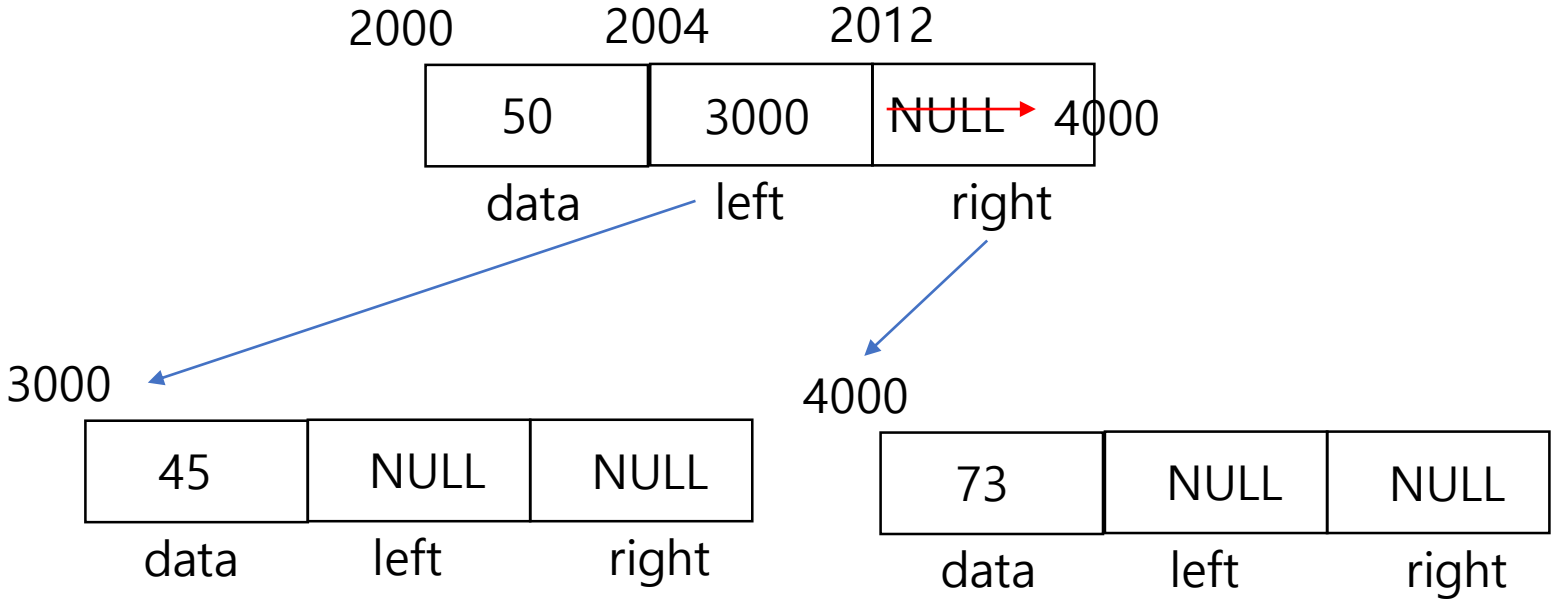


root의 주소에 접근하니 NULL값이므로 조건문을 실행한다

root의 주소에 할당된 데이터와 비교하여 그 값보다 작으면 left의 주소를 가지고 재귀호출을 한다








```
tree *get_node(){

        tree *tmp;
        tmp = (tree *)malloc(sizeof(tree));
        tmp -> right = NULL;
        tmp -> left = NULL;
        return tmp;

}

void tree_ins(tree **root,int data){

        if(*root == NULL){
                *root = get_node();
                (*root)->data=data;
                return;
        }else if((*root)->data > data)
                tree_ins(&(*root)->left,data);
        else if((*root)->data < data)
                tree_ins(&(*root)->right,data);
}
```

주소에 접근했을 때 NULL을 만나면 그 자리에 데이터를 할당한다

root의 주소에 할당된 데이터와 비교하여
그 값보다 크면 right의 주소를 가지고
재귀호출을 한다

(main)

1000 | 2000

root

tree_ins

1000	32
root	data

tree_ins

2004	32
root	data

tree_ins

3004	32
root	data

```
void tree_ins(tree **root,int data){

        if(*root == NULL){
                *root = get_node();
                (*root)->data=data;
                return;
        }else if((*root)->data > data)
                tree_ins(&(*root)->left,data);
        else if((*root)->data < data)
                tree_ins(&(*root)->right,data);

}
```

2000 2004 2012

50	3000	4000
data	left	right

3000 3004 3012

45	NULL → 5000	NULL
data	left	right

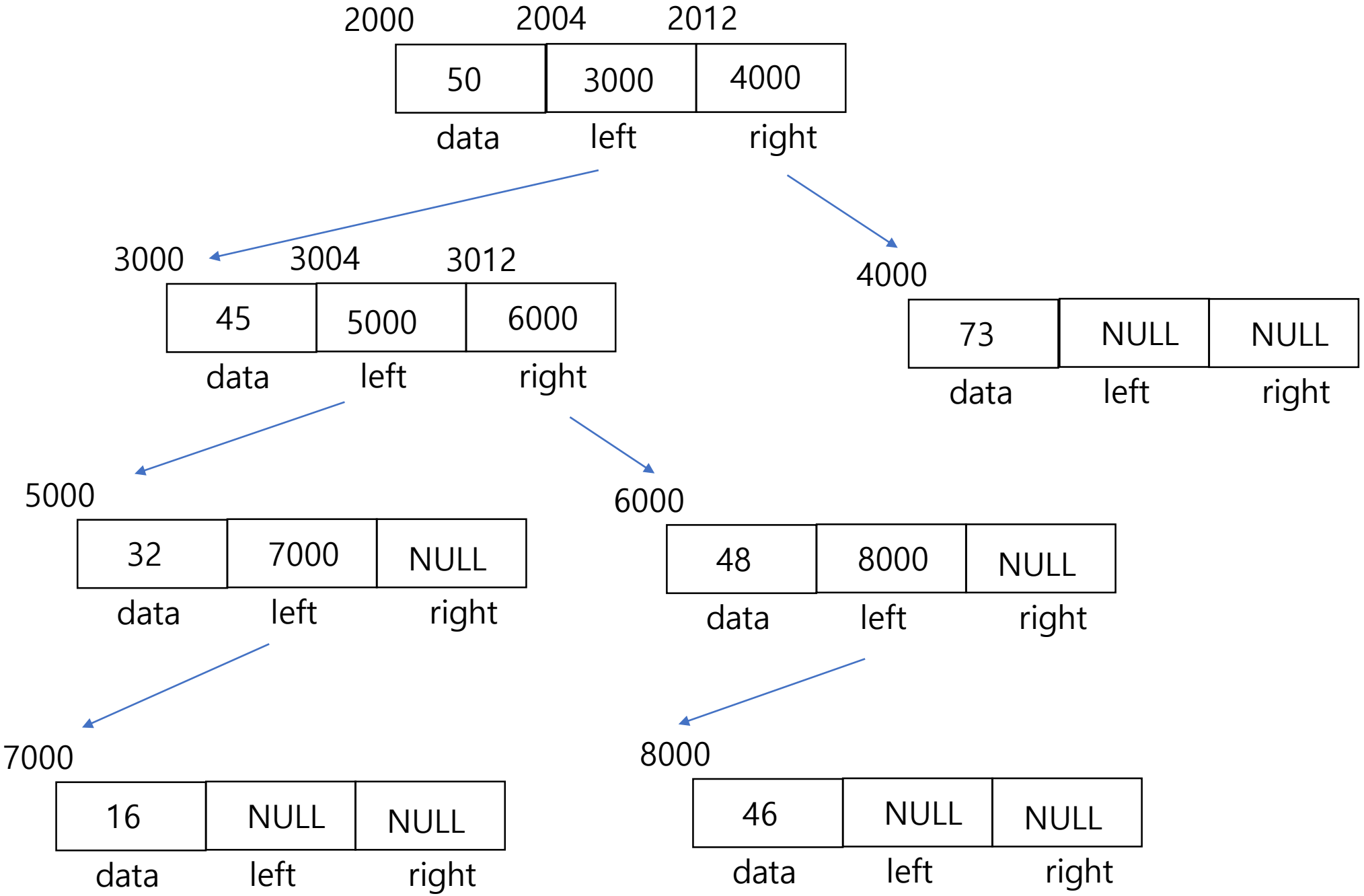
4000

73	NULL	NULL
data	left	right

5000

32	NULL	NULL
data	left	right

이런 방식으로 7번째 까지 가면


2000
2004
2012
50
3000
4000
data
left
right
3000
3004
3012
45
5000
6000
data
left
right
4000
73
NULL
NULL
data
left
right
5000
32
7000
NULL
data
left
right
6000
48
8000
NULL
data
left
right
7000
16
NULL
NULL
data
left
right
8000
46
NULL
NULL
data
left
right

print_tree

```c
int main(void){

        int i;
        int data[14] = {50,45,73,32,48,46,16,37,120,47,130,127,124};

        tree *root = NULL;

        for(i=0; data[i]; i++)
                tree_ins(&root,data[i]);

        print_tree(root);

        /*delete_tree(root,50);
          printf("after delete\n");

          print_tree(root);*/

        return 0;
}
```

```
mhn@mhn-900X3L:~/my_proj/c/9_s$ ./a.out
data = 50, left = 45, right = 73
data = 45, left = 32, right = 48
data = 32, left = 16, right = 37
data = 16, left = NULL, right = NULL
data = 37, left = NULL, right = NULL
data = 48, left = 46, right = NULL
data = 46, left = NULL, right = 47
data = 47, left = NULL, right = NULL
data = 73, left = NULL, right = 120
data = 120, left = NULL, right = 130
data = 130, left = 127, right = NULL
data = 127, left = 124, right = NULL
data = 124, left = NULL, right = NULL
mhn@mhn-900X3L:~/my_proj/c/9_s$
```

```c
void print_tree(tree *root){

        if(root){
                printf("data = %d, ",root->data);

                if(root->left)
                        printf("left = %d, ",root->left->data);
                else printf("left = NULL, ");

                if(root->right)
                        printf("right = %d\n",root->right->data);
                else printf("right = NULL\n");

                print_tree(root->left);
                print_tree(root->right);
        }

}
```

root주소값에 접근하여 data를 출력한다

각 data의 왼쪽과 오른쪽에 어떤 값이 있는지 알 수 있다.

왼쪽으로 내려가며 먼저 출력한 뒤 오른쪽으로 이동하여 찍는다