TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018.03.07
10 일차

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 신민철
akrn33@naver.com

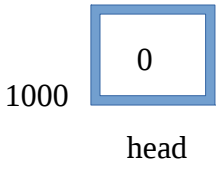
Queue 연결리스트

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<time.h>
#define EMPTY 0

struct node{
	int data;
	struct node * link;
};

typedef struct node queue;

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

void enqueue(queue** head, int data){

	if(*head == NULL)
	{
		*head = get_node();
		(*head)->data = data;
		return;
	}
```

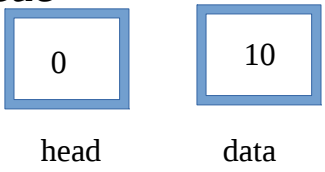
```
        enqueue(&(*head)->link,data);

}

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;

}

void print_queue(queue* head){
        queue* tmp = head;

        while(tmp)
        {
                printf("tmp->data = %d\n",tmp->data);
                printf("tmp->link = %p\n",tmp->link);
                tmp = tmp->link;
        }
}
```

```
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;
}
```

Main

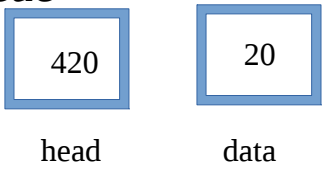


Enqueue

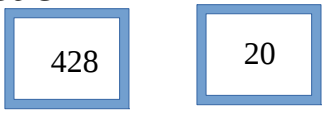


Get_node

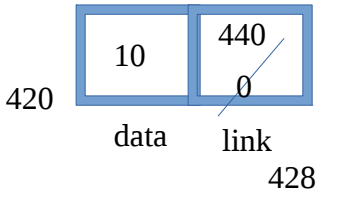
Enqueue

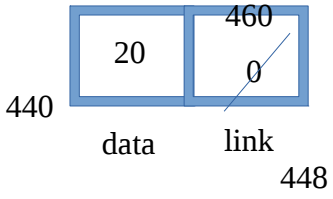


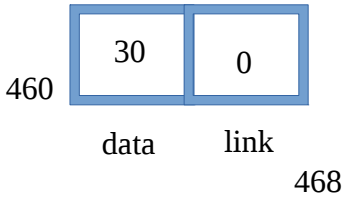
Enqueue

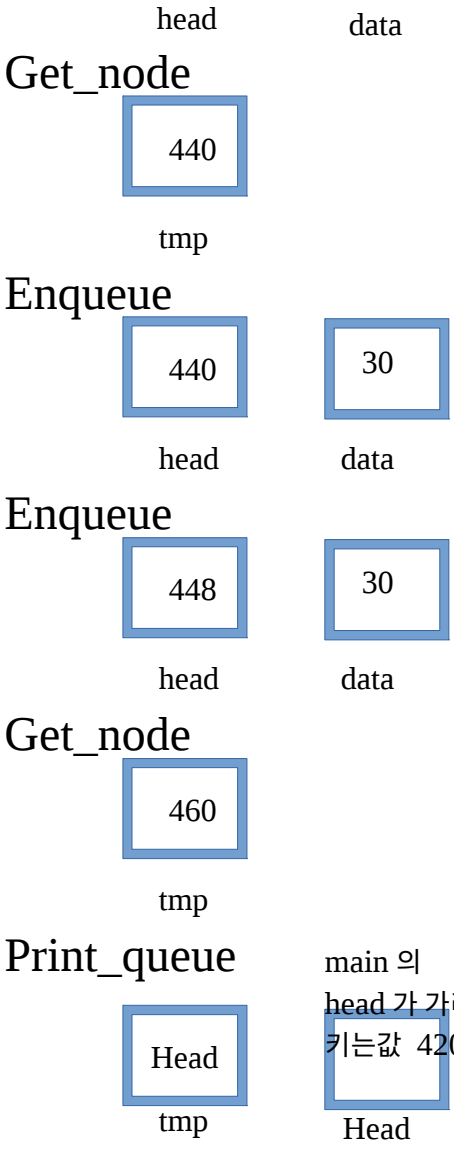


Heap







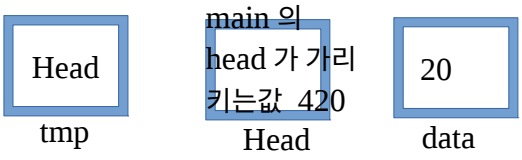


출력결과

tmp = head = 0x727420
tmp->data = 10
tmp->link = 0x727440
tmp->data = 20
tmp->link = 0x727460
tmp->data = 30
tmp->link = (nil)

While(tmp 가 0 이아니면)
Printf(tmp 의값은 주소가 들어있는데 주소안에 가리키는 data)
(tmp→data)
tmp = tmp→link tmp 가 가리키는곳의 link 를 tmp 에 대입한다
(연결된곳의 다음을 알려준다)
루프 While 을 돌면서 tmp 의 가리키는값이 없어지면 종료한다.

Dequeue



인자로 들어온 data 가 tmp 의 data 와 같은가?

Yes No

data 삭제후
삭제된 data 의
가리키는곳을
삭제된 data 를
가리키는
tmp 가 가리킴

다음링크에
있는 값을
비교한다.

출력결과는
tmp = head == 0x727420
tmp->data = 10
tmp->link = 0x727460
tmp->data = 30
tmp->link = (nil)

위의 출력을 보면 0x727440 이 사라지고 그 데이터도 사라지면서
420 의링크 428 이 460 을 가리키게 하고 사라진 것을 볼 수 있다.

알 수 있다.

Tree

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#define EMPTY 0
struct node{
	int data;
	struct node* left;
	struct node* right;
};

typedef struct node tree;

tree* chg_node(tree* root)
{
	tree* tmp = root;

	if(!root->right)
		root = root->left;
	else if(!root->left)
		root = root->right;

	free(tmp);

	return root;
}
tree* get_node()
{
	tree* tmp;
	tmp = (tree*)malloc(sizeof(tree));
```

```
	return tmp;
}
tree* find_max(tree* root, int* data)
{
	if(root->right)
		root->right = find_max(root->right, data);
	else
	{
		*data = root->data;
		root = chg_node(root);
	}
	return root;
}

tree* delete_tree(tree* root, int data)
{
	int num;
	tree* tmp;
	if(root == NULL)
	{
		printf("Not Found\n");
		return NULL;
	}
	else if(root->data > data)
		root->left = delete_tree(root->left, data);
	else if(root-data < data)
		root->right = delete_tree(root->right, data);
	else if(root->left && root->right)
	{
		root->left = find_max(root->left, &num);
		root->data = num;
	}
	else
		root = chg_node(root);
	return root;
```

```
}
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);
	}
}


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);
```

```
}

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;
}
```

insert 하고 출력한 결과를 그림으로 그린결과는 이러한 모양이 된다.