Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정

#14

강사 : Innova Lee(이 상훈)

학생 : 김 시윤

1. 배운내용 복습

1)Stack.
저번 과제때 스택의 그림을 그렸다.
오늘 수업시간에 그 그림을 토대로
소스코드를 작성하는 연습을 하였고,
오늘은 집에서 소스코드를 완벽히 구연한 출력 결과까지 구현해 보았다.
스택의 소스코드는 첫시간에 외운게 있어서 그렇게
어렵지 않았다.

하지만 프린트 함수를 구현하려 했지만
구현하지 못하였다.

```
siyun@siyun-CR62-6M:~/hw14$ vi stack.c
siyun@siyun-CR62-6M:~/hw14$ ./a.out
 30,  20,  10, stack is EMPTY 0,
siyun@siyun-CR62-6M:~/hw14$
```

siyun-CR62-6M: ~/hw14

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

#define EMPTY 0

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

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

// push -> pop -> print

void push(stack **top,int data)
{
        stack *tmp;
        tmp = *top;
        *top=get_node();
        (*top)->data=data;
        (*top)->link=tmp;
}
int pop(stack **top)
{
        stack *tmp;
        int num;
        tmp=*top;
        if(*top==EMPTY)
        {
        printf("stack is EMPTY");
        return 0;
        }
        num=(*top)->data;
        (*top)=(*top)->link;
        free(tmp);
        return num;
}

int main(void)
{
        stack *top=EMPTY;
```

2)queue

stack에서 출력순서만 바꾸면 queue이다

queue를 소스코드를 보지않고 그림을 그려 구현해보았다.

구현 후 원래의 소스코드와 비교를 해보았는데 매우 비슷했다.

아마 스택의 효과인 것 같다.

```
\siyun@siyun-CR62-6M:~/hw14$ gcc queue.c
siyun@siyun-CR62-6M:~/hw14$ ./a.out
10 , 20 , 30 ,
siyun@siyun-CR62-6M:~/hw14$ vi queue.c
siyun@siyun-CR62-6M:~/hw14$
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <malloc.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)
{
        queue *tmp;
        tmp = *head;
        if(*head == NULL)
        {
        *head=get_node();
        (*head)->data=data;
        return;
        }
        enqueue(&(*head)->link,data);
}

void print_queue(queue *head)
{

        queue *tmp;
        tmp=head;
        while(tmp)
        {
        printf("%d , ",tmp->data);
        tmp=tmp->link;
        }
}

int main(void)
{
        queue *head=NULL;
        enqueue(&head,10);
        enqueue(&head,20);
        enqueue(&head,30);
        print_queue(head);
        printf("\n");
```

3.delete queue

delete queue가 제일 구현하기 쉬웠다.

이유는 queue에서 delete 함수만 추가하면 끝이기 때문인거같다.

```
siyun@siyun-CR62-6M:~/hw14$ vi dequeue.c
siyun@siyun-CR62-6M:~/hw14$ ./a.out
10 , 30 ,
siyun@siyun-CR62-6M:~/hw14$
```

main은 위의 queue와 동일하고

delete queue 함수만 추가하였다.

```
}

void delete_queue(queue **head,int data)
{
        queue *tmp;
        int num;
        tmp=*head;
        if((*head)==EMPTY)
        {       printf("There is no data your queue");
                return;
        }
        if((*head)->data != data)
        {
                delete_queue(&(*head)->link,data);
        }
        else if((*head)->data == data)
        {
                (*head)=(*head)->link;
                num=(*head)->data;
                free(tmp);
        }

}

void print_queue(queue *head)
```

4. tree

tree는 insert를 받을 때 큐랑 동일하지만
왼쪽루트 오른쪽루트가 나눠지므로 크기비교가 들어간다.
또 프린트할 때 전위식 출력을 했기 때문에
왼쪽부터 읽고 오른쪽으로 읽게 재귀함수를 설정하였다.

```
siyun@siyun-CR62-6M:~/hw14$ vi tree.c
siyun@siyun-CR62-6M:~/hw14$ gcc tree.c
siyun@siyun-CR62-6M:~/hw14$ ./a.out
50 ,45 ,32 ,38 ,42 ,48 ,46 ,73 ,101 ,120 ,124 ,
50 , 45 , 32 , 38 , 42 , 48 , 46 , 73 , 101 , 120 , 124 ,
siyun@siyun-CR62-6M:~/hw14$
```

print 함수를 두 개를 만들었다.
더블포인터와 포인터의 개념을 잡기위한 것으로
인자가 더블포인터인 함수와
인자가 싱글포인터인 함수로 두 개의 프린트 함수를 도전해보았다.
소스코드가 길어 표에다 쓰도록한다.

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

#define EMPTY 0

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->left=EMPTY;
        tmp->right=EMPTY;
        return tmp;

}
void insert_tree(tree **root,int data)
{
        tree *tmp;
        tmp=*root;
        if(*root==NULL)
        {
                *root=get_node();
                (*root)->data=data;
                return;
        }
        if((*root)->data > data)
        {       insert_tree(&(*root)->left,data);
        }
        if((*root)->data < data)
        {
insert_tree(&(*root)->right,data);
        }
}
```

```
void print_tree(tree *root)
{
        tree *tmp;
        tmp=root;
        if(tmp!=EMPTY)
        {printf("%d , ",tmp->data);
        print_tree(root->left);
        print_tree(root->right);
        }
}

void print_tree2(tree **root)
{
        tree *tmp;
        tmp=*root;
        if(tmp !=EMPTY)
        {
        printf("%d , ",tmp->data);
        print_tree2(&(*root)->left);
      }
}

int main(void)
{
        tree *root=EMPTY;
        int i;
int a[20] =
{50,60,40,75,45,30,128,35,25,125};

        for(i=0;i<a[i];i++)
        {
                insert_tree(&root,a[i]);
        }
        print_tree(root);
        printf("₩n");
        print_tree2(&root);
        printf("₩n");
return 0;
}
```