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

강사 – Innova Lee(이상훈)

gcccompil3r@gmail.com

학생 – hoseong Lee(이호성)

hslee00001@naver.com

- 자료구조

- Stack
  a) 정의
  b) 예제
- Queue
  a) 정의
  b) 예제

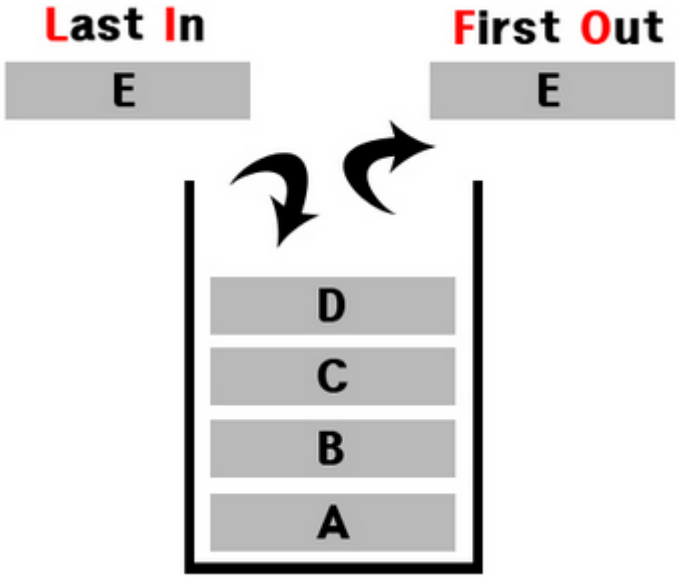
자료 구조란?

컴퓨터에 자료를 효율적으로 어떻게 저장할 것인가?
메모리 절약효과와 자료를 저장하는 데 수행하는 시간도 줄일 수 있다.

선형구조 – 큐, 리스트, 스택
비선형구조 - 트리,그래프

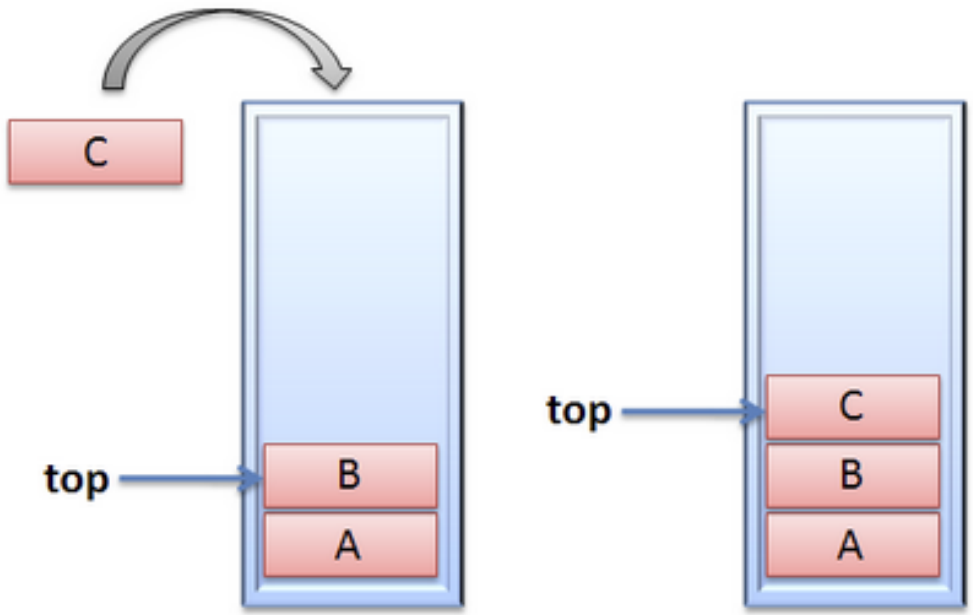
스택(stack)이란?

자료를 한쪽으로 보관하고 꺼내는 **LIFO(Last In First Out)** 자료구조. 스텍에 자료를 보관하는 연산을 **push** 라 말하고, 꺼내는 연산을 **pop** 이라 말한다. 가장 최근에 보관한 위치정보를 **Top or 스택포인터**라 한다.



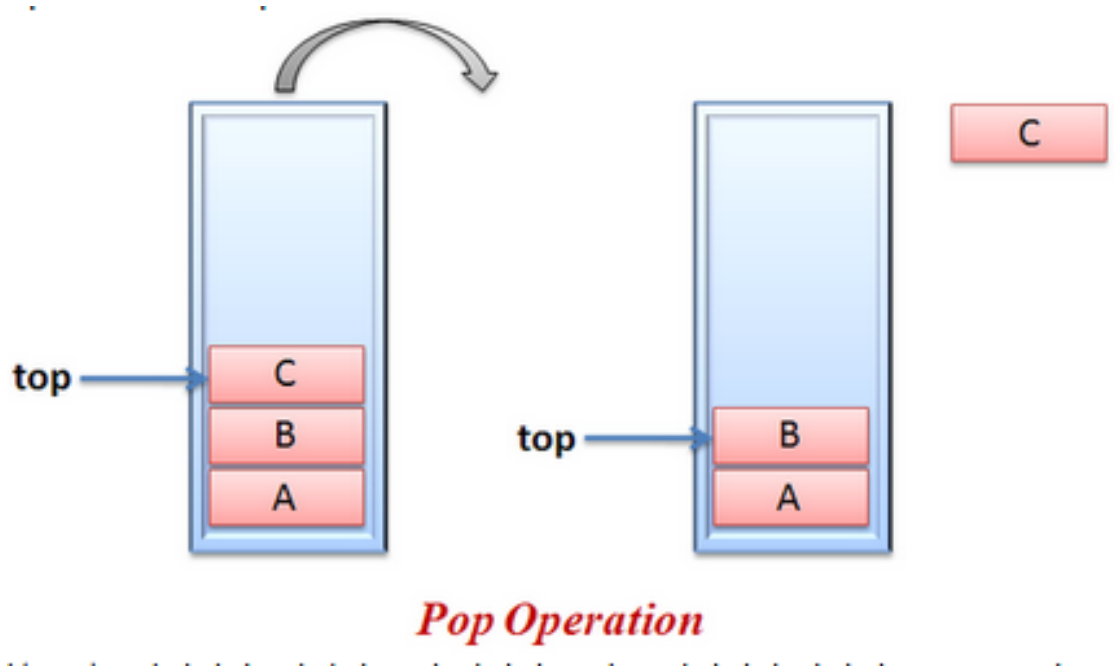
→ 프링글스

push



Push Operation

pop

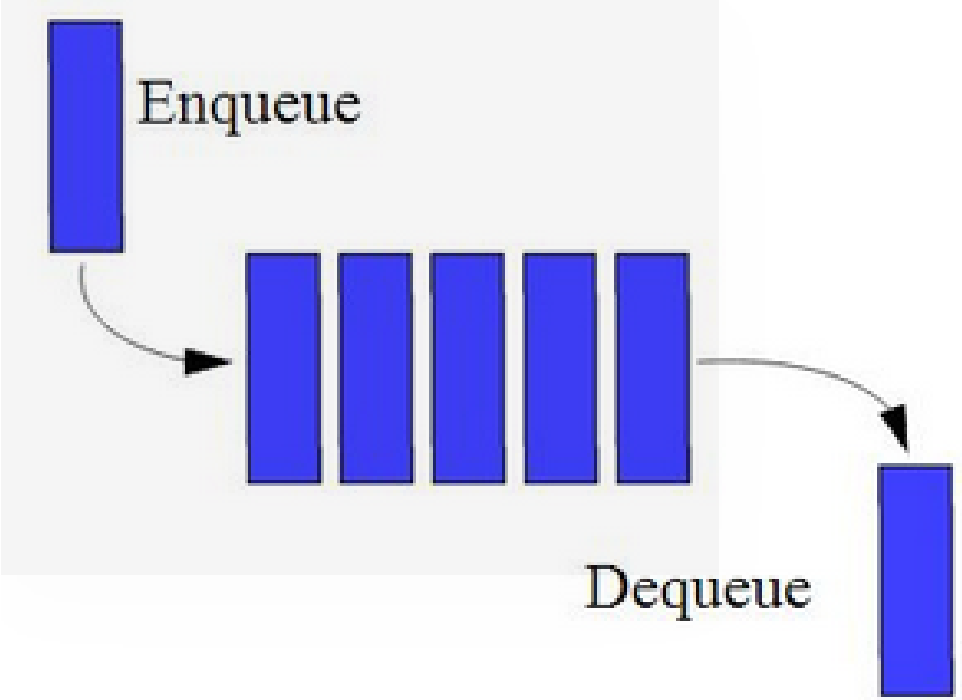


```
#include <stdio.h>
#include <malloc.h>

#define EMPTY 0

struct node{                                    // 구조체 node 선언
        int data;                               //  int 형 data 변수 선언
        struct node *link;                      //  구조체 포인터 link 변수 선언 ( stack 을 가르킬수 있다.)
};
typedef struct node Stack;                      // 구조체 node → Stack 별칭으로 선언

Stack *get_node()                               // 구조체 포인터 get_node 함수 선언
{
        Stack *tmp;                             // 구조체 포인터  tmp 변수 선언
        tmp=(Stack *)malloc(sizeof(Stack));     // (stack 형)의 동적메모리를 (구조체 stack)의 크기만큼 할당하고,  tmp 에 반환하는 주소값 저장
        tmp→link=EMPTY;                         // 구조체 포인터 tmp 안에 link 에 접근하여 0 을 넣어라.
        return tmp;                             //  tmp 반환. → 할당된 메모리 주소값 반환
}

void push(Stack **top, int data)                // 반환값이 없는 push 함수 생성,  **top, int 형 data 변수 선언
{
        Stack *tmp;                             // 구조체 포인터 선언 ( tmp 는 stack 을 가르킬 수 있다)
        tmp = *top;                             // push 함수의 top 변수가 가르키는 main top 의 값을  tmp 변수에 넣어라.
        *top = get_node();                      // push 함수의 top 변수가 가르키는 곳에 get_node()함수의 리턴값을 넣어라.
        (*top)->data = data;                    // *top 이 가르키고 있는 할당된 heap memoy 의 (tmp) data 에 접근하여 data 값을 넣어라.
        (*top)->link = tmp;                     // “
```

```
}
int pop(Stack **top)                    // 반환값 int 형을 가진 pop 함수 생성, 이중포인터 top 변수 선언
{       Stack *tmp;                     // 구조체 포인터 변수 선언, tmp
        int num;                        // int 형 변수 선언, num
        tmp = *top;                     // pop 함수의 top 변수가 가르키는 곳의 값을 tmp 변수에 넣어라.
        if(*top == EMPTY)
        {
                printf("Stack is empty!!!\n");
                return 0;
        }
        num = tmp→data;                 //  구조체 포인터 tmp 변수가 가르키는 곳 안에 data 에 접근
        *top = (*top)→link;             //  구조체포인터 top 변수가 가르키는 곳 안의 data 에 접근
        free(tmp);                      //  동적 메모리 해제
        return num;
}
int main(void)
{
        Stack *top = EMPTY;    // top          // main 의 구조체포인터 변수 생성 ( top 은 stack 을  가르킬 수 있다.)
        push(&top, 10);
        push(&top, 20);
        push(&top, 30);
        printf("%d\n", pop(&top));
        printf("%d\n", pop(&top));
        printf("%d\n", pop(&top));
        printf("%d\n", pop(&top));
        return 0;
}
```

큐란?

큐는 자료를 한쪽으로 보관하고 다른쪽에서 꺼내는 **FIFO(First In First Out)** 방식의 자료구조이다. 큐에 자료를 보관하는 연산을 **ENQUEUE** or PUT 이고, 꺼내는 연산을 **DEQUEUE** or GET 라고 말한다. 그리고 보관할 위치 정보를 **tail** or rear, 꺼낼 위치 정보를 **head** or front 라고 말한다.



쌤꺼.

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

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

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

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

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


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

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;
        }
}
```

```
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;
}
```

내꺼 → 진행중

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

#define EMPTY 0

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

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

void ENQUEUE(Queue **head,int data)
{
   Queue *tail;
   tail= *head;
   *head = get_node();
   (*head)->data = data;
   (*head)->link = tail;
}
```

```
int main(void)
{
  Queue *head = EMPTY;
  ENQUEUE(&head,10);
  return 0;
}
```