TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 최대성
c3d4s19@naver.com

2018.03.05. – 008일차 수업

---

* 자료구조 스택(Stack)

스택(Stack)은 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료구조이다.

스택은 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출 방식으로 자료를 처리한다.

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

typedef int Data;
typedef struct _node{
    Data data;
    struct _node* link;
}Stack;

Stack* getNode(){
    Stack* newNode =
(Stack*)malloc(sizeof(Stack));
    return newNode;
}

void pushStack(Stack** top, Data data){
    Stack* tmp;
    tmp = getNode();
    tmp->link = *top;
    tmp->data = data;
    *top = tmp;
}

Data popStack(Stack** top){
    if((*top) == NULL){
        printf("Stack is empty!₩n");
        return 0;
    }
    Stack* tmp = *top;
    Data data = (*top)->data;
    *top = (*top)->link;
    free(tmp);
    return data;
}

void printAllOfStack(Stack** top)
{
    Stack* tmpNode = *top;
    while(tmpNode != NULL)
    {
        printf("%d₩n", tmpNode->data);
        tmpNode = tmpNode->link;
    }
}
```

```c
int main(){
    Stack* Top;
    Top = NULL;
    pushStack(&Top,10);
    pushStack(&Top,20);
    pushStack(&Top,20);
    pushStack(&Top,20);

    printAllOfStack(&Top);

    printf("₩n₩n");

    printf("%d₩n",popStack(&Top));
    printf("%d₩n",popStack(&Top));
    printf("%d₩n",popStack(&Top));
    printf("%d₩n",popStack(&Top));
    printf("%d₩n",popStack(&Top));

    return 0;
}
```

---

* 자료구조 큐(Queue)

큐(Queue)는 먼저 삽입된 자료가 먼저 나오는 선입선출 방식으로 자료를 처리하고 한쪽 끝으로 자료를 삽입하고 반대쪽 끝에서 삭제 작업이 이루어진다.

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

typedef int Data;
typedef struct _node{
    Data data;
    struct _node* link;
}Queue;

void enqueue(Queue** head, Data data){

    Queue* newQueue;
    if(*head == NULL){
        *head = (Queue*)malloc(sizeof(Queue));
        (*head)->link = NULL;
        (*head)->data = data;
    }
    else{
        enqueue( &((*head)->link), data);
        //tmp = tmp->link;
    }
}

Data dequeue(Queue** head){
    Queue* tmp = *head;
```

```
    Data data;

    if(tmp == NULL){
        printf("Queue is empty!₩n");
        return 0;
    }
    data = (*head)->data;
    *head = (*head)->link;
    free(tmp);
    //printf("%d₩n",data);
    return data;
}

void printAllOfQueue(Queue** top)
{
    Queue* tmpNode = *top;
    while(tmpNode != NULL)
    {
        printf("%d₩n", tmpNode->data);
        tmpNode = tmpNode->link;
    }
}

Data delQueue(Queue** head, Data delData){

    Queue* tmp = *head;
    Data data;
    if(*head == NULL){
        printf("Have no data: %d!₩n",delData);
        return 0;
    }
    else if( (*head)->data == delData ){
        data = (*head)->data;
        *head = (*head)->link;
        free(tmp);
        printf("del data: %d!₩n",delData);
        return data;
    }
    else{
        delQueue( &((*head)->link), delData);
    }
}


int main(){

    Queue* head;
    head = NULL;

    enqueue(&head,10);
    enqueue(&head,20);
    enqueue(&head,30);
    enqueue(&head,40);
    enqueue(&head,50);
    enqueue(&head,60);
    enqueue(&head,70);
    enqueue(&head,80);
    enqueue(&head,90);

    delQueue(&head,40);
    delQueue(&head,40);
    delQueue(&head,50);
    delQueue(&head,60);

    delQueue(&head,10);
    printAllOfQueue(&head);


    return 0;
}
```

---

* 이진트리 데이터 넣는 함수

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

typedef int Data;
typedef struct _tree{
    Data data;
    struct _tree* Right;
    struct _tree* Left;
}Tree;

Tree* getNewTree(){
    Tree* newTree = (Tree*)malloc(sizeof(Tree));
    newTree->Right = NULL;
    newTree->Left = NULL;
    return newTree;
}
void insertDataToTree(Tree** tree, Data data){
    Data cmpData;
    if( *tree == NULL ){
        *tree = getNewTree();
        (*tree)->data = data;
    }
    else if( data <= (*tree)->data ){
        insertDataToTree( &((*tree)->Left),
data );
    }
    else{
        insertDataToTree( &((*tree)->Right),
data );
    }
}
int main(){
    Tree* tree = NULL;
    Data d[13] = { 50, 45, 73, 32, 48, 46, 16, 37,
120, 47, 130, 127, 124 };
    for(int i = 0; i < 13; i++){
        insertDataToTree(&tree, d[i]);
    }
    return 0;
}
```