TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

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

2018.05.21. – 057 일차 수업

계산기 프로그램(미완성)

<stack.h 헤더파일>

```
#ifndef __STACK_H__
#define __STACK_H__

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

typedef char* Data;

typedef struct _node{
    Data data;
    struct _node* next;
}Node;

typedef struct _stack{
    int num_of_data;
    struct _node* head;
}Stack;

Node* get_node(void);
Stack* get_stack(void);
void ins_stack(Stack* stack, Data data);
Data pop_stack(Stack* stack);
Data peek_stack(Stack* stack);
void print_stack(Stack* stack);

#endif
```

<stack.c 파일>

```
#include "stack.h"

Node* get_node(void){
    Node* newNode =
(Node*)malloc(sizeof(Node));
    newNode->next = NULL;
    return newNode;
}

Stack* get_stack(void){
    Stack* newStack =
(Stack*)malloc(sizeof(Stack));
    newStack->num_of_data = 0;
    newStack->head = NULL;
    return newStack;
}

void ins_stack(Stack* stack, Data data){
    Node* tmp = stack->head;
    stack->head = get_node();
    stack->head->next = tmp;
    stack->head->data = data;
    stack->num_of_data++;
}

Data pop_stack(Stack* stack){
    if(!stack->head)
        return NULL;
    Data pop = stack->head->data;
    Node* tmp = stack->head->next;
    free(stack->head);
    stack->head = tmp;
    stack->num_of_data--;
    return pop;
```

```c
}


Data peek_stack(Stack* stack){
    if(!stack->head)
        return NULL;
    return stack->head->data;
}


void print_stack(Stack* stack){
    Node* tmp = stack->head;
    for(int i = 0; i < stack->num_of_data; i+
+){
        printf("%s  ",tmp->data);
        tmp = tmp->next;
    }
    printf("\n");
}



#define TEST 0
#if TEST


int main(){
    char str1[] = "1st string";
    char str2[] = "2nd string";
    char str3[] = "3rd string";
    char str4[] = "4th string";
    char str5[] = "5th string";


    Stack* testStack = get_stack();


    ins_stack(testStack,(Data)str1);
    ins_stack(testStack,(Data)str2);
    ins_stack(testStack,(Data)str3);

    print_stack(testStack);


    printf("%s\n", pop_stack(testStack));
    printf("%s\n", peek_stack(testStack));


    print_stack(testStack);


    ins_stack(testStack,(Data)str4);
    ins_stack(testStack,(Data)str5);


    print_stack(testStack);
}


#endif
```

<InfixToPostfix.h 헤더파일>

```c
#ifndef __INFIX_TO_POSTFIX_H__
#define __INFIX_TO_POSTFIX_H__
#define DATA_MAX_NUM 32
#define INFIX_MAX_LEN 128
#include <stdio.h>
#include <string.h>
#include <ctype.h>


char* get_data(char* string);
char** sep_infix_data(char* infix);
int op_priority(char op);
char** infix_to_postfix(char** infix);
void print_expr(char** expr);


#endif
```

<InfixToPostfix.c 파일>

```c
#include "InfixToPostfix.h"
#include "stack.h"


//string 동적 할당
char* get_data(char* string){
    char* newStr = (char*)malloc(strlen(string + 1));
    strncpy(newStr, string, strlen(string + 1));
    return newStr;
}


//중위표기법 의미별 string 분리
char** sep_infix_data(char* infix){
    int i = 0, s = 0, index = 0;
    char* tmp = (char*)malloc(sizeof(char) * INFIX_MAX_LEN);
    char** sep_data = (char**)malloc(sizeof(char*) * DATA_MAX_NUM);


    sep_data[index++] = &tmp[0];
    while(1){
        if(!infix[i]){
            sep_data[--index] = NULL;
            break;
        }
        //공백이면 넘어감
        else if(infix[i] == ' '){
            ;
        }
        //문자인 경우
        else if(!isdigit(infix[i])){
            tmp[s++] = infix[i];
            tmp[s++] = '\0';
            sep_data[index++] = &tmp[s];
        }
        //숫자이고 다음 char 도 숫자인 경우
        else if(isdigit(infix[i + 1])){
            tmp[s++] = infix[i];
        }
        //마지막 숫자인 경우
        else{
            tmp[s++] = infix[i];
            tmp[s++] = '\0';
            sep_data[index++] = &tmp[s];
        }
        i++;
    }
    return sep_data;
}


//연산자 우선순위 반환 함수
int op_priority(char op){
    switch(op){
        case '(':
        case ')':
            return 10;
        case '+':
        case '-':
            return 5;
        case '*':
        case '/':
            return 3;
        case '^':
            return 1;
        default:
            return -1;
    }
}
```

```c
//중위표기법 -> 후위표기법 변환 함수
char** infix_to_postfix(char** infix){
    int pf_idx = 0, if_idx = 0;
    int op_cmp1, op_cmp2;
    char* tmp;
    char** postfix = (char**)malloc(sizeof(char*) * DATA_MAX_NUM);
    Stack* stack = get_stack();


    while(1){
        //NULL 인 경우
        if(infix[if_idx] == NULL){
            while(stack->num_of_data > 0){
                postfix[pf_idx++] = pop_stack(stack);
            }
            break;
        }
        //숫자인 경우
        else if(isdigit(*infix[if_idx])){
            postfix[pf_idx++] = infix[if_idx++];
        }
        // ')' 괄호인 경우
        else if(*infix[if_idx] == ')'){
            while( *(tmp = pop_stack(stack)) != '('){
                postfix[pf_idx++] = tmp;
            }
            if_idx++;
        }
        //연산자인 경우
        else{
            //stack 에 아무것도 없는 경우
            if(!peek_stack(stack)){
                op_cmp1 = op_priority(*infix[if_idx]);
                op_cmp2 = 10;
            }
            else{
                op_cmp1 = op_priority(*infix[if_idx]);
                op_cmp2 = op_priority(*(peek_stack(stack)));
            }


            //새로운 연산자 우선순위가 낮은 경우
            if(op_cmp1 > op_cmp2){


                while(!(stack->num_of_data == 0 || *peek_stack(stack) == '(')){
                    postfix[pf_idx++] = pop_stack(stack);
                }
            }
            //새로운 연산자 우선순위가 높은 경우
            else{
                ins_stack(stack, infix[if_idx++]);
            }
        }
    }
    return postfix;
}


//연산식 print
void print_expr(char** expr){
    for(int i = 0; expr[i] ; i++){
        printf("%s ", expr[i]);
    }
    printf("\n");
}
```

```
#define TEST 1
#if TEST
int main(){
    char infix1[INFIX_MAX_LEN] = "( 11+ 222 * 33) ^21";
    char infix2[INFIX_MAX_LEN] = " 11 + 222 * (33 ^ 21)";


    char** result1 = sep_infix_data(infix1);
    char** result2 = infix_to_postfix(result1);


    print_expr(result1);
    print_expr(result2);
}
#endif
```