2020-2
Week6

# 데이터구조및실습

## 6주차 실습

# Program 연습1-1

//기본배열이용한 스택

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

#define MAX_STACK_SIZE 100      // stack의 최대크기 지정
typedef int element;            // 원소는 int로 지정
element stack[MAX_STACK_SIZE];  // 1차원 int 배열
int top = -1;                   // 초기상태 top은 -1

void error(const char X[]) {    // 오류 메시지 출력
    fprintf(stderr, X);
}

bool is_full(void) {
    return (top == MAX_STACK_SIZE - 1); // top이 최대면 true 반환
}
bool is_empty(void) {
    return (top == -1);                 // top이 -1이면(빈 스택) true 반환
}
```

# Program 연습1-2

//기본배열이용한 스택

```
element pop(void) {
    if (is_empty()) {
        error("Stack empty");
        exit(1);
    }
    else return stack[top--];// top 원소값 반환 후 top 위치 조정
}

element peek(void) {
    if (is_empty()) {
        error("Stack empty");
        exit(1);
    }
    else return stack[top];     // top원소값 반환 하나 원소는 그대로 남아 있음
}

void push(element item) {
    if (is_full()) {
        error("Stack Overflow");
        return;
    }
    else stack[++top] = item;   // top 증가 후 원소값 저장
}

int main(void)
{
    push(1);
    push(2);
    push(3);
    printf("%d₩n", pop());
    printf("%d₩n", pop());
    printf("%d₩n", pop());
    printf("%d₩n", pop()); // 스택 empty 상황
    return 0;
}
```

```
3
2
1
Stack empty
```

# Program 연습2-1

//구조체 이용한 스택구현

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

#define MAX_STACK_SIZE 100
typedef int element;

typedef struct StackType{
    element data[MAX_STACK_SIZE];   // 배열의 요소는 element 타입
    int top;                        // top 값을 member로 포함함
}StackType;

void init_stack(StackType *s) {     // 인자로서 구조체 포인터를 전달받음
    s->top = -1;
}

void error(const char X[]) { // 오류 메시지 출력
    fprintf(stderr, X);
}

bool is_full(StackType *s) {
    return (s->top == (MAX_STACK_SIZE - 1)); // top이 최대면 true 반환
}

bool is_empty(StackType *s) {
    return (s->top == -1); // top이 -1이면(빈 스택) true 반환
}
```

# Program 연습2-2

//구조체 이용한 스택구현

```
element pop(StackType *s) {
    if (is_empty(s)) {
        error("Stack empty");
        exit(1);
    }
    else return s->data[(s->top)--];    // top이 가리키는 원소값 반환 후 top 위치 조정(top--)
}

void push(StackType *s, element item) {
    if (is_full(s)) {
        error("Overflow");
        return;
    }
    else s->data[++(s->top)] = item;    // top을 1 증가 후 해당 위치에 원소값 저장
}

int main(void)
{
    StackType s;    // 구조체 변수 s 선언
    init_stack(&s); // 구조체 초기화
    push(&s,1);
    push(&s,2);
    push(&s,3);
    printf("%d₩n", pop(&s));
    printf("%d₩n", pop(&s));
    printf("%d₩n", pop(&s));
    printf("%d₩n", pop(&s)); // 스택 empty 상황
    return 0;
}
```

```
3
2
1
Stack empty
```

# Program 연습3

//구조체 동적할당

```
int main(void)
{
    StackType *s;    // 구조체 포인터변수 s 선언
    s = (StackType*)malloc(sizeof(StackType)); // 동적 메모리 할당
    init_stack(s); // call by reference
    push(s, 1);
    push(s, 2);
    push(s, 3);
    printf("%d₩n", pop(s));
    printf("%d₩n", pop(s));
    printf("%d₩n", pop(s));
    printf("%d₩n", pop(s)); // 스택 empty 상황
    free(s);                // 메모리 반환 필요
    return 0;
}
```

# Program 연습4-1

//구조체멤버 동적할당

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

typedef int element;
typedef struct StackType {
    element *data; // 스택을 동적으로 할당하고 이를 data가 가리키도록 함
    int capacity;  // 현재 동적할당받은 메모리 크기
    int top;
}StackType;

void init_stack(StackType *s) {
    s->capacity = 1;
    s->top = -1;
    s->data = (element*)malloc(s->capacity*sizeof(element)); // 1개 요소 저장공간 우선 동적할당
}

void error(const char X[]) { // 오류 메시지 출력
    fprintf(stderr, X);
}

bool is_full(StackType *s) {
    return (s->top == s->capacity - 1); // top이 최대면 true 반환
}

bool is_empty(StackType *s) {
    return (s->top == -1); // top이 -1이면(빈 스택) true 반환
}
```

# Program 연습4-2

//구조체멤버 동적할당

```
element pop(StackType *s) {
    if (is_empty(s)) {
        error("Stack empty");
        exit(1);
    }
    else return s->data[(s->top)--];// top 원소값 반환 후 top 위치 조정
}

void push(StackType *s, element item) {
    if (is_full(s)) {
        s->capacity *= 2;  // capacity가 부족하면 두배로 확장
        s->data = (element*)realloc(s->data, s->capacity * sizeof(element));
        //  현재 내용을 유지하면서 동적 메모리를 다시 할당
    }
    s->data[++(s->top)] = item; // top 증가 후 원소값 저장
}

int main(void)
{
    StackType s;    // 구조체 변수 s 선언
    init_stack(&s); // call by reference
    push(&s, 1);
    push(&s, 2);
    push(&s, 3);
    printf("%d₩n", pop(&s));
    printf("%d₩n", pop(&s));
    printf("%d₩n", pop(&s));
    printf("%d₩n", pop(&s)); // 스택 empty 상황
    free(s.data);
    return 0;
}
```

# Program 연습5-1

//연결리스트로 스택 구현

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

typedef int element;

typedef struct StackNode{ // 스택노드 구조체
    element data;
    struct StackNode*link;
}StackNode;

typedef struct {
    StackNode*top;      // 스택 top 노드주소
}LinkedStackType;

void init(LinkedStackType *s) {
    s->top = NULL;      // 초기에는 빈 stack
}

void error(const char X[]) { // 오류 메시지 출력
    fprintf(stderr, X);
}

bool is_full(LinkedStackType *s) {
    return 0;               // full이 될 수 없음
}

bool is_empty(LinkedStackType *s) {
    return (s->top == NULL); // top이 NULL이면(빈 스택) true 반환
}
```

# Program 연습5-2

//연결리스트로 스택 구현

```
element pop(LinkedStackType *s) {
    if (is_empty(s)) {
        error("Stack empty");
        exit(1);
    }
    else {
        StackNode*temp = s->top;     // 포인터 temp가 삭제할 노드를 가리킴
        element data = temp->data;   // 반환할 값을 저장
        s->top = s->top->link;       // top이 가리키던 원소가 가리키는 link가 새로운 top이 됨
        free(temp);                  // 삭제된 노드가 사용하던 메모리 반환
        return data;                 // top이 기존에 가리키던 원소값 반환
    }
}

void push(LinkedStackType *s, element item) {
    StackNode*temp = (StackNode*)malloc(sizeof(StackNode)); // 새로 추가할 노드 생성
    temp->data = item;          // 데이터 저장
    temp->link = s->top;        // 새로 생성된 노드의 link가 기존의 top을 가리킴
    s->top = temp;              // 새로 생성된 노드가 새로운 top이 됨
}

void print_stack(LinkedStackType*s) {
    for (StackNode*p = s->top; p!= NULL; p=p->link)
        printf("%d -> ", p->data);
    // top에서부터 링크를 이동하면서 NULL이 아닐동안 출력
    printf("NULL ₩n"); // 마지막 링크는 NULL
}
```

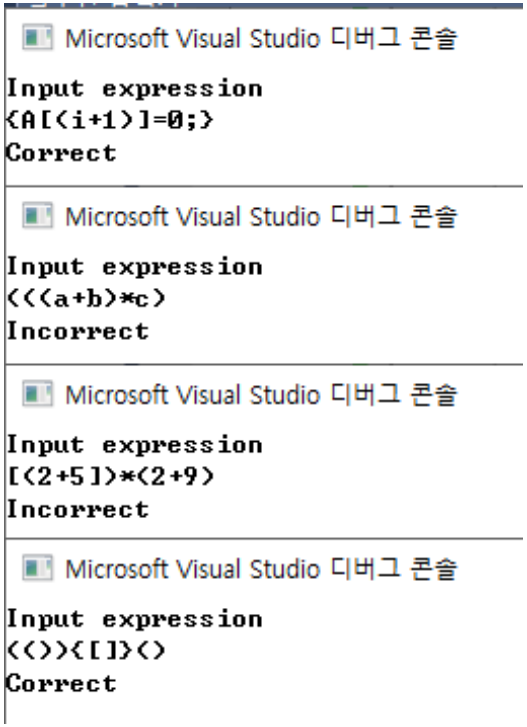
```
int main(void)
{
    LinkedStackType s ;    // 동적 스택 헤드포인터 선언
    init(&s); // call by reference
    push(&s, 1);    print_stack(&s);
    push(&s, 2);    print_stack(&s);
    push(&s, 3);    print_stack(&s);
    pop(&s);  print_stack(&s);
    pop(&s);  print_stack(&s);
    pop(&s);  print_stack(&s);
    pop(&s);  print_stack(&s);// 스택 empty 상황
    return 0;
}
```

# Program 연습6-1

//괄호쌍 검사

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_STACK_SIZE 100

typedef char element;  // element는 char 타입
typedef struct stack {
    element data[MAX_STACK_SIZE];
    int top;
}StackType;
```



# Program 연습6-2

//괄호쌍 검사

```
bool parenth_check(char *X) {
    StackType*s = (StackType*)malloc(sizeof(StackType));// 열린 괄호 담을 스택 포인터 선언
    char open_ch, ch;
    init_stack(s);  // 스택 초기화
    for (int i = 0; i < strlen(X); i++) { // 최대 문자열 길이만큼 반복
        ch = X[i];          // 문자를 읽어옴
        switch (ch) {
        case '(':   case '[':   case '{':
            push(s, ch);    // 읽은 문자가 열린 괄호면 push
            break;
        case ')':   case ']':   case '}':   // 읽은 문자가 닫힌 괄호면
            if (is_empty(s)) return false;  // 닫힌 괄호인데 pop할 게 없으면 false 반환
            else {
                open_ch = pop(s); // 읽은 문자가 닫힌 괄호면 스택에서 pop한 문자와 쌍이 맞는지 체크
                if (((open_ch == '(') && (ch != ')')) ||
                    ((open_ch == '{') && (ch != '}')) ||
                    ((open_ch == '[') && (ch != ']')))
                    return false;  // 괄호쌍이 안맞으면 바로 false
            }
            break;
        } // end of switch
    } // end of for
    if (!is_empty(s)) return false; // 문자열 체크 끝났는데 스택에 열린 괄호가 남아있으면 실패
    return true; // 위 체크를 다 거치고 나면 성공
}

int main(void)
{
    char str[100];
    printf("Input expression₩n");
    gets_s(str);
    if (parenth_check(str)) printf("Correct₩n");
    else printf("Incorrect₩n");
    return 0;
}
```

# Program 연습7-1

//후위표기식 계산

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

#define MAX_STACK_SIZE 100
typedef int element;

typedef struct StackType{
    element data[MAX_STACK_SIZE]; // 피연산자, 즉 정수만 스택에 넣을 거니까 int 배열로 선언
    int top;
}StackType;
```

# Program 연습7-2

```
int eval(char *X) {
    int op1, op2;           // 피연산자
    char ch;                // 문자열에서 읽어온 문자
    StackType s;            // 구조체 포인터 선언
    init_stack(&s);         // 구조체 초기화

    for (int i = 0; i < strlen(X) ; i++) { // 문자열의 길이만큼 반복해서 체크
        ch = X[i];          // 문자 읽어오기                                    //후위표기식 계산
        if (ch != '+' && ch != '-' && ch != '*' && ch != '/')  // 4칙연산자 아니면
            push(&s, ch-'0');  // 피연산자(operand)이므로 문자를 숫자로 바꾼 후 push
        else {  // 연산자(operator)이면 이 전 operand 두 개를 pop 해서 연산 수행
            op2 = pop(&s);
            op1 = pop(&s);
            switch (ch) {
            case '+': push(&s, op1 + op2); break; // 연산 수행 결과를 다시 push
            case '-': push(&s, op1 - op2); break;
            case '*': push(&s, op1 * op2); break;
            case '/': push(&s, op1 / op2); break;
            default:
                error("expression error₩n"); // 4칙연산자 이외의 기호인 경우 오류
            } // end of switch
        } // end of else
    } // end of for
    return pop(&s); // 연산 결과를 pop해서 반환
}

int main(void)
{
    char X[100];
    printf("후위표기식 입력(정수와 사칙연산으로만 구성)₩n");
    gets_s(X);
    printf("result: %d₩n", eval(X));
    return 0;
}
```

# Program 연습8-1

// 중위식→후위식 변환

```
element peek(StackType *s) {
    if (is_empty(s)) {
        error("Stack empty");
        exit(1);
    }
    else return s->data[s->top]; // top원소값 반환
}

int pri(char ch) {  // 연산자 우선순위 체크
    switch (ch) {
    case '(': case ')': return 0;
    case '+': case '-': return 1;
    case '*': case '/': return 2;
    }
    return -1;
}
```

# Program 연습8-2

// 중위식→후위식 변환

```
void infix_to_postfix(char *X) {
    int op;                     // 피연산자
    char ch, top_op;            // 문자열에서 읽어온 문자
    StackType s;                // 구조체 포인터 선언
    init_stack(&s);             // 구조체 초기화

    for (int i = 0; i < strlen(X); i++) {       // 문자열의 길이만큼 반복해서 체크
        ch = X[i];                              // 문자 읽어오기
        switch (ch) {
        case '+': case '-': case '*': case'/':  // 연산자인경우
            while (!is_empty(&s) && (pri(peek(&s)) >= pri(ch))) { // 스택에 다른 연산자가 쌓여있으면
                printf("%c", pop(&s));          // 우선순위 높은 연산자까지는 모두 pop해서 출력
            }
            push(&s, ch);                       // 우선순위 높은 연산자까지 모두 pop 한 후 push
            break;
        case '(':                               // 여는 괄호면 바로 push
            push(&s, ch);
            break;
        case ')':                               // 닫는 괄호면
            top_op = pop(&s);
            while (top_op != '(') {             // 여는 괄호까지 스택에 쌓여있는 연산자를 모두 pop해서 출력
                printf("%c", top_op);
                top_op = pop(&s);
            }
            break;
        default:
            printf("%c", ch);                   // 피연산자의 경우 그냥 출력
        } // end of switch
    }  // end of for
    //while (s.top >= 0) printf("%c", pop(&s));  // 스택에 남아있는 연산자를 모두 pop하면서 출력
    while (!is_empty(&s)) printf("%c", pop(&s));  // 스택에 남아있는 연산자를 모두 pop하면서 출력
}
```

# Program 연습8-3

// 중위식→후위식 변환

```
int main(void)
{
    char X[100];
    printf("Input an infix expression₩n");
    gets_s(X);
    printf("<result>₩n");
    infix_to_postfix(X);
    printf("₩n");
    return 0;
}
```

```
Input an infix expression
(2+3)*4+9
<result>
23+4*9+
```