자료구조(DS)

<과제1>

제공된 동영상과 프로그램 파일을 참고하여 배열을 이용하여 구현한 스택 프로그램을 만들 수 있는 능력을 키우세요.

(연결 리스트는 이번 과제가 아님)

배열을 이용하여 구현한 스택 프로그램 제작 및 제작 능력 개발

한라대학교 ICT 공학융합부

정보통신소프트웨어학과

201932030

송현교

목차

1) C 언어에서 구현할 수 있는 가장 간단한 코드

구현 및 구현 모습:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    return 0;
}
```

```
Microsoft Visual Studio 디버그 콘솔

C:₩Users₩군인₩source₩repos₩Project7₩x64₩Debug₩Project7.exe(프로세스 25644개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

알게 된 내용:

C 언어에서 코드를 작성할때 " 이것은 공식이다" 라 생각하며
항상 이용하던 틀 같은 존재였지만
코드는 작동하지 않으면 아무런 의미가 없다고 생각했기에
이렇게 단순한 상태로만 작성하고 끝내 본 적은
이번이 처음이었습니다.
이 형태에서 #define _CRT_SECURE_NO_WARNINGS 과
#include <stdio.h> 가 무엇을 의미하는 걸까? 라는 궁금증이 생겼고,
궁금을 해결하기 위해 찾아 본 결과
#define _CRT_SECURE_NO_WARNINGS 은
호환성 문제가 발생할 여지가 있는 scanf 함수 등을
_s 를 붙이지 않고 그냥 사용하고 싶을 때
자잘한 경고가 발생하는 것을 막아주는 함수라는 것을 알게 되었고,
#include <stdio.h> 는
우리가 주로 사용하는 printf 등의 함수를 사용하기 위해
<stdio.h> 내에 정의되어 있는 많은 함수들을 내 코드 내에
포함(include) 시켜 주겠다,
라는 의미인 것을 다시금 알게 되었습니다.
가장 단순화된 코드이기에, 복학 후 C 언어에 발을 딛기
어려움을 느끼고 있던 저에게 간단하고 쉬운 방향으로
C 언어에 접근할 수 있는 계기가 되어 주었습니다

2) 스택 배열 구현1 (기초적인 스택 구조 이해하기)

구현 및 구현 모습:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define StackLen 5 // 5라는 숫자를 앞으로 StackLen 이라고 정의합니다.
typedef char data; // char타입을 생성하는 char의 이름을 data로 정의합니다.

struct Stack {
    data Arr[StackLen]; // 만약 위에서 typedef를 하지 않았다면, = char Arr[5]
    int top; // 정수type top 을 생성합니다.
}S;

void stack() // 스택이라는 함수에 대해,↓실행
{
    S.top = -1; //top 에 -1을 대입합니다.
}

int main() // 프로그램이 시작되면 main 이라는 함수를 먼저 찾아 시작하기 때문에, int main에 포함된 stack(); 가 실행되어 먼저 스택이 생성됩니다.
{
    stack(); // 스택을 생성합니다.
    return 0; // int main은 int를 반환하는 함수이기 때문에 정수값을 반환해 줘야 해서 return 0으로 0을 반환합니다.
}
```

```
선택 Microsoft Visual Studio 디버그 콘솔

C:₩Users₩군인₩source₩repos₩Project7₩x64₩Debug₩Project7.exe(프로세스 34772개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

알게 된 내용:

지금 배우고 있는 Stack 구조도 생소했지만,

#define, typedef 은 완전히 처음 보는 명령어들이었습니다.

#define 을 통해 위 코드처럼 5를 StackLen으로 등의 정의하는 법을 습득하고, typedef를 통해 타입 변환법 또한 알게 되었습니다.

또한,

Stack을 생성함에 있어 struct 함수가 의미하는 것을 익혀

캐릭터 타입의Arr[5]; 과 정수 타입의"top" 를 포함하는
S 라는 이름의 구조체를 직접 만들 수 있는 능력을 개발할 수 있었습니다.

3) 스택 배열 구현2 (생성한 배열이 공백인지 아닌지 if 문을 이용하여 알아보자)

구현 및 구현 모습:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define StackLen 5 // 5라는 정수를 StackLen이라는 이름으로 정의합니다.
typedef char data; //char 타입 생성자의 이름을 data로 정의합니다.

struct Stack { // 구조체를 생성하겠습니다.
    data Arr[StackLen]; // ① 위에서 만약 typedef를 하지 않았다면 char Arr[5]와 같습니다.
    int top; // ② 정수타입의 top 을 생성합니다.
}S; // ①과 ②를 가진 S라는 이름의 구조체를 생성했습니다.

void Stack() {
    S.top = -1;
}

int IsEmpty()
{
    if (S.top <= -1) // 만약 S라는 구조체에서 top이 -1보다 작거나 같으면,
        return 1; // (유효데이터가 하나도 없으면) 1을 반환 (공백 상태이다.) 합니다.
    else // S.top이 -1보다 크면,
        return 0; // (유효데이터가 존재하면) 0을 반환 (공백 상태가 아니다. 합니다.
}

int main()
{
    Stack(); // Stack를 생성합니다.

    if (IsEmpty()) // 만약 정수 타입의 IsEmpty에서 1이 반환되었으면,
        printf("₩n공백 상태 입니다."); // ("₩n공백 상태입니다.") 를 출력합니다.
    else // 만약 정수 타입의 IsEmpty에서 0이 반환되었으면,
        printf("₩n공백 상태가 아닙니다."); // ("₩n공백 상태가 아닙니다.") 를 출력합니다.

    return 0; // int main은 정수타입이므로 0을 반환합니다.
}
```

```
Microsoft Visual Studio 디버그 콘솔

공백 상태 입니다.
C:₩Users₩군인₩source₩repos₩Project7₩x64₩Debug₩Project7.exe(프로세스 33972개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

알게 된 내용:

S.top 에서 즉, top 이 유한데이터의 범위를 나타낸다는 것을 지난 대면수업때 교수님께 배웠습니다.

본 프로그램에서는 if 문을 사용하여 S.top 이 -1보다 작거나 같으면,

1을 반환(공백 상태임) 하고 그렇지 않을 시 0을 반환(공백 상태가 아님) 하는데,

수업때 배운 "top 이 -1 이면 유한데이터가 하나도 없다" 라는 내용을

실제로 코드를 작성해가며 확실하게 알게 되었습니다.

4) 스택 배열 구현3 (배열에 내가 원하는 값을 삽입해보자)

구현 및 구현 모습 :

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define StackLen 5
typedef char data;

struct Stack {
    data Arr[StackLen];
    int top;
}S;

void stack() {
    S.top = -1;
}

int IsEmpty() {
    if (S.top <= -1)
        return 1;
    else
        return 0;
}

void showStackData() { // showStackData 라는 이름의 함수를 생성하겠습니다.
    printf("₩n<stack> : "); // ①, 줄바꿈 후에 <stack> : 를 출력합니다.
    for (int a = 0; a <= S.top; a++) // 정수 타입의 a가 구조체 S의 top(배열 주소)보다 작으면, a에 1씩 증가시키는 것을 반복합니다..
        printf("%c", S.Arr[a]); // ① 뒤에, 구조체 S에서 Arr 배열의 a주소에 있는 값을 char 타입으로 불러와서 출력합니다.
}

void push(data e) { // char 타입의 e를 배열에 집어넣는 push라는 이름의 함수를 생성하겠습니다.
    if (S.top >= StackLen - 1) { // 만약, S구조체에서 top의 값이 4 이상이면,
        printf("₩n포화 상태 입니다! 값을 더 이상 넣을 수 없습니다."); // 배열에서의 창고는 총 5칸이고 그렇다면 top의 최댓값이 4이기 때문에 값을 더 이상 넣을 수 없음을 알려줍니다.
    }
    S.top++; // 배열에 e를 하나 삽입하므로, (방에 물건을 하나 넣었으므로) top의 값이 1 추가됩니다.
}

int main() {
    int num; // 정수 타입의 num을 생성합니다.
    char val; // 캐릭터 타입의 val을 생성합니다.

    stack(); // 스택을 생성합니다.

    while (1) {
        printf("₩n< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : ");
        do {
            scanf_s("%d", &num); // 정수를 입력받아 num 값에 대입합니다.
            while (getchar() != '₩n'); //버퍼에 남아 있는 자료 없애기 (이 코드만 교수님께서 올려주신 자료에서 그대로 가져왔습니다.)
        } while (num < 1 || num>3);// num이 1보다 작거나 3보다 크면, 다시↑

        switch (num) {
        case 1:printf("삽입할 값을 입력하시오.");
            scanf_s("%c", &val); // char 타입으로, 문자 하나를 입력받아 val 에 대입합니다.
            push(val); // val에 대입된 값을 배열에 (방에) 집어넣습니다.
            break;
        case 2: printf("삭제합니다.");
            break;
        case 3: printf("프로그램을 종료합니다.");
            return 0;
        }
    }
}
```

```
C:₩Users₩군인₩source₩repos₩Project7₩x64₩Debug₩Project7.exe

< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 1
삽입할 값을 입력하시오.a

<stack> : a
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 1
삽입할 값을 입력하시오.a

<stack> : aa
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 :
```

알게 된 내용:

push()를 통해 내가 원하는 값을 배열에 삽입할 수 있는 법을 배웠으며, while(getchar() != '\n\); 이라는 명령을 알 수 있었습니다.

2 번 코드부터 스택의 내용을 단계적으로 습득할 수 있게끔 과제를 제시하여 주셨는데,

이제 제가 원하는 스택을 작성할 수 있는 발판이 만들어 지고 있는 듯한 느낌이 듭니다.

그리고 현재 이 4 번 코드까지 작성해오면서 느낀 것은 자료구조만이 아닌 프로그램에서도,

IsEmpty() 나showStackData(), push() 처럼 내가 먼저 특정 역할을 수행할 함수를 만들고,

그렇게 구성해 둔 함수들을 마지막 main 함수에서 가져다만 쓰면

프로그래밍을 매우 효율적으로 할 수 있을 것 같습니다.

5) 스택 배열 구현 4 (배열에서의 삭제 삽입 방법을 습득해보자)

구현 및 구현 결과:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define StackLen 5
typedef char data;

struct Stack {
    data Arr[StackLen];
    int top;
}S;

void stack( ) {
    S.top = -1;
}

int IsEmpty( ) {
    if (S.top <= -1)
        return 1;
    else
        return 0;
}

void showStackData( ) {
    printf("₩n<stack> : ");
    for (int a = 0; a <= S.top; a++)
        printf("%c", S.Arr[a]);
}

void push(data e) {
    if (S.top >= StackLen - 1) {
        printf("₩n포화 상태 입니다! 값을 더 이상 넣을 수 없습니다.");
    }
    S.top++;
    S.Arr[S.top] = e;
    showStackData( );
}

data pop( ) { // pop 이라는 함수를 생성하겠습니다.
    if (S.top <= -1) { // 만약 , top이 -1보다 작거나 같으면, 즉 유한데이터가 존재하지 않을 시 ↓
        printf("₩n공백 상태입니다 !값을 더 이상 지울 수 없습니다."); // 공백 상태이며 더 이상 빼낼 유한 데이터가 없다는 것을 알려줍니다.
        return '₩n'; // '₩n'을 반환합니다.
```

(전역 범위) main()

```c
int main() {
    int num;
    data val;

    stack();

    while (1) {
        printf("₩n< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : ");
        do {
            scanf_s("%d", &num);
            while (getchar() != '₩n');
        } while (num < 1 || num>3);

        switch (num) {
        case 1:printf("삽입할 값을 입력하시오.");
            scanf_s("%c", &val);
            push(val);
            break;
        case 2: {
                printf("₩n%c가 삭제되었습니다.", pop());
            break;
        }
        case 3: printf("프로그램을 종료합니다.");
            return 0;
        }
    }
}
```

0 2 ↑ ↓ 검색 심도:

값

```
C:₩Users₩군인₩source₩repos₩Project7₩x64₩Debug₩Project7.exe

< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 1
삽입할 값을 입력하시오.a

<stack> : a
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 1
삽입할 값을 입력하시오.b

<stack> : ab
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 1
삽입할 값을 입력하시오.c

<stack> : abc
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 2

<stack> : ab
c가 삭제되었습니다.
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 2

<stack> : a
b가 삭제되었습니다.
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 : 2

<stack> :
a가 삭제되었습니다.
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 프로그램 종료 :
```

알게 된 내용:

pop() 을 이용하여 창고에서 물건을 빼내는 구조를 알게 되었고,
생성 할 수 있게 되었습니다.
다만 코드를 전체적으로 보았을 때는 pop 이 어디에 어떻게 작용하는지를 이해 할 수 있었습니다만
pop 을 생성하는 과정에서 이해하기 어려운 부분이 있어 다음 수업 시간때 질문 드리고자 합니다.

6) 스택 배열 구현5 (peek() 를 이용하여 창고의 가장 위에 있는 물건을 꺼내보자)

구현 및 구현 결과:

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define StackLen 5
typedef char data;

struct Stack {
    data Arr[StackLen];
    int top;
}S;

void stack() {
    S.top = -1;
}

int IsEmpty() {
    if (S.top <= -1)
        return 1;
    else
        return 0;
}

void showStackData() {
    printf("₩n<Stack> : ");
    for (int a = 0; a <= S.top; a++) {
        printf("%c", S.Arr[a]);
    }
}

void push(data e) {
    if (S.top >= StackLen - 1) {
        printf("₩n포화 상태입니다 ! 더 이상 값을 넣을 수 없습니다.");
    }
    S.top++;
    S.Arr[S.top] = e;
    showStackData();
}

data pop() {
    if (S.top <= -1) {
        printf("₩n공백 상태입니다 ! 더 이상 값을 지울 수 없습니다.");
        return '₩n';
    }
    S.top--;
    showStackData();
    return S.Arr[S.top + 1];
}

data peek() {
    if (S.top <= -1) { // 만약 , top이 -1보다 작거나 같으면, 즉 유한데이터가 존재하지 않을 시 ↓
        printf("₩n공백 상태입니다 ! 유한데이터가 없기 때문에 값을 찾을 수 없습니다.");  // 유한데이터가 없는 상태이기 때문에 맨 위의 값을 찾을 수 없음을 알려줍니다.
    }
    showStackData();
    return S.Arr[S.top];  // Arr 배열에서, top 주소에 있는 값을 반환합니다.
}
```

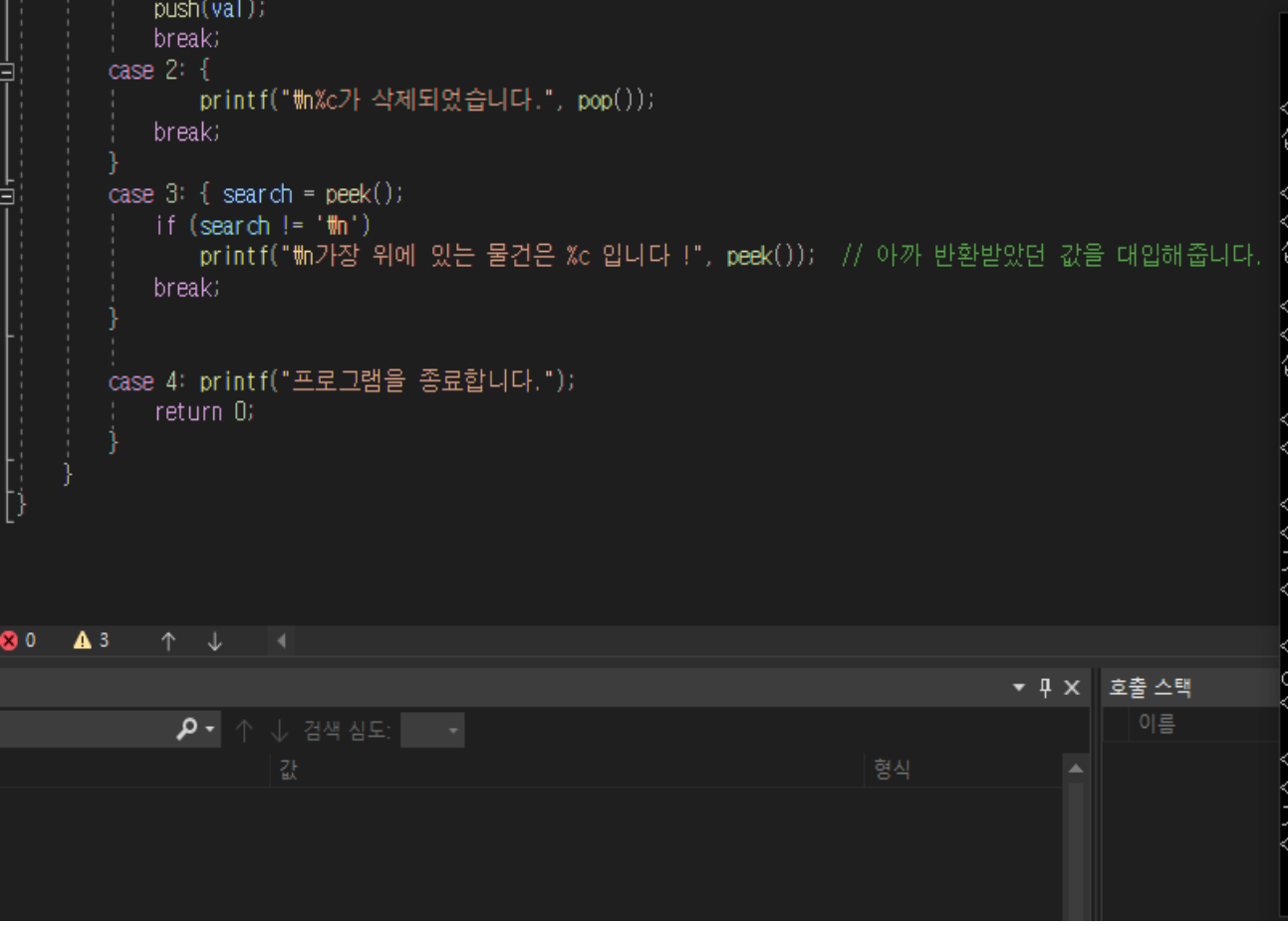
```c
int main() {
    int num;
    char val, search;

    stack();

    while (1) {
        printf("₩n< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : ");
        do {
            scanf_s("%d", &num);
            while (getchar() != '₩n');
        } while (num < 1 || num>4);

        switch (num) {
        case 1:printf("삽입할 값을 입력하시오.");
            scanf_s("%c", &val);
            push(val);
            break;
        case 2: {
                printf("₩n%c가 삭제되었습니다.", pop());
            break;
        }
        case 3: { search = peek();
            if (search != '₩n')
                printf("₩n가장 위에 있는 물건은 %c 입니다 !", peek());  // 아까 반환받았던 값을 대입해줍니다.
            break;
        }

        case 4: printf("프로그램을 종료합니다.");
            return 0;
        }
    }
}
```




```
C:₩Users₩군인₩source₩repos₩Project7₩x64₩Debug₩Project7.exe

< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : 1
삽입할 값을 입력하시오.a

<Stack> : a
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : 1
삽입할 값을 입력하시오.b

<Stack> : ab
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : 1
삽입할 값을 입력하시오.c

<Stack> : abc
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : 3

<Stack> : abc
<Stack> : abc
가장 위에 있는 물건은 c 입니다 !
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : 2

<Stack> : ab
c가 삭제되었습니다.
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 : 3

<Stack> : ab
<Stack> : ab
가장 위에 있는 물건은 b 입니다 !
< Stack > 1. 자료 삽입 2. 자료 삭제 3. 자료 조회 4. 프로그램 종료 :
```

알게 된 내용:

peek()를 이용하여 내가 넣은 값들 중 가장 위에 있는 값을 찾는 프로그램을 코딩할 수 있게 되었습니다.

peek 도 pop과 마찬가지로 조금 이해가 덜 가는 부분이 있어, 수업을 하는 날 만나뵙고 질문을 드리고자 합니다.

결과 및 검토, 과제와 관련하여 하고 싶은 말 :

이렇게 총 6 개의 프로그램을 코딩하면서, 앞에서 어렵게 느꼈던 반복문도 되짚어 보고
넘어갈 수 있는 기회가 되었고,
순차적으로 프로그램을 제시해 주셨기 때문에
차근차근 익혀가면서 올라오게 될 수 있었습니다.
1 을 쌓지 않은 상태에서 2 를 쌓을려고 하는 것은 욕심이다, 라고 말씀 해 주신게
이 과제를 통해 더욱 이해가 갔습니다.
만약 순차적이 아니라, 처음부터 6 번째 코딩을 시도했다면 도저히 하지 못 했을 것 같습니다.
제가 작성한 프로그램이 이렇게 체계적으로 작동하는 것을 직접 본다는 경험이 매우 신선하고,
기초라 말씀하셨던 이 배열을 잘 할 수 있게 되면,
빨리 다른 상위의 프로그램들도 작성을 시도해 보고 싶습니다.
과제 확인 해 주시느라 고생 많으셨습니다. 감사합니다 !!