

[Source]

[stack.h]

```
#define StackElement int
#define MaxStackSize 10
#define boolean unsigned char
#define true 1
#define false 0

typedef struct {
    int top;
    StackElement data[MaxStackSize];
}Stack;
```

[stack.c]

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include "stack.h"

void StackCreate( Stack *S )
{
    int i;
    S->top = -1;
    for ( i=0; i<MaxStackSize; i+ + )
    {
        S->data[i] = 0;
    }
    return;
}

void Push( Stack *S, StackElement newElement )
{
    if ( S->top+ 1 == MaxStackSize )
    {
        printf ( "WnData is FULL and DonW't insert dataWn" );
        return;
    }
    S->top+ +;
    S->data[S->top] = newElement;

    printf( "Wn" );
    return;
}

int Pop( Stack *S )
{
    int res;
    if ( S->top == -1 )
    {
        printf ( "WnData no Exist and DonW't pop dataWn" );
        return;
    }
    res = S->data[S->top];
    S->data[S->top] = 0;
    S->top--;

    printf( "Wn" );
    return res;
}
```

```

boolean StackEmpty( Stack *S )
{
    printf ( "Wn" );
    if ( S->top == -1 )
        return true;
    else
        return false;
}

boolean StackFull( Stack *S )
{
    printf ( "Wn" );
    if ( S->top+ 1 == MaxStackArraySize )
        return true;
    else
        return false;
}

void StackClear( Stack *S )
{
    printf ( "Wn" );
    StackCreate(S);
    return;
}

void StackShowStructure( Stack *S )
{
    int i;
    if ( !S )
    {
        printf ( "WnStack not createWn" );
        return;
    }
    if ( S->top == -1 )
    {
        printf ( "WnStack is EmptyWn" );
        return;
    }
    printf ( "Wn" );
    for ( i=0; i<=S->top; i+ + )
        printf ( "%d", S->data[i] );
    printf ( "Wn" );
    return;
}

void Operate(Stack *testStack, char cmd)
{
    int tempFirst, tempSecond, tempResult;
    if(StackEmpty(testStack) == false)
    {
        tempFirst = Pop(testStack);
        tempSecond = Pop(testStack);
        switch(cmd) {
            case '+' :
                tempResult = tempSecond + tempFirst;
                break;
            case '-' :
                tempResult = tempSecond - tempFirst;
                break;
            case '*' :
                tempResult = tempSecond * tempFirst;
                break;
        }
    }
}

```

```

        }
        printf("Result of operation : %d\n", tempResult);
        Push(testStack, tempResult);
    }
    else
        printf("WaThere is only one element in the Stack!\n");
}

int main() {
    Stack *testStack;
    char cmd; //명령 입력받을 변수, 입력받을 숫자는 무조건 한 자리 숫자만 가능하다.
    int x;

    printf("***** Choose Command!! *****\n");
    printf("+: Add, -: Sub, *: Multiply\n");
    printf("C: StackClear, S: ShowStructure, Q: Quit\n");
    printf("*****\n");

    testStack = ( Stack * )malloc( sizeof(Stack) );
    memset( testStack, 0, sizeof(Stack) ); // 할당된 메모리를 0으로 초기화
    StackCreate(testStack);

    do {
        printf("Command(0 ~ 9, +, -, *): ");
        cmd = getch();
        putchar( cmd );
        cmd = toupper( cmd );

        if (isdigit(cmd)) {
            x = (int)(cmd-'0');
            Push(testStack, x);
        }
        else {
            switch (cmd) {
                case '+' : case '-' : case '*': // Operate
                    Operate(testStack, cmd);
                    break;
                case 'C' : case 'c' : // Clear Memory
                    StackClear(testStack);
                    break;
                case 'S' : case 's' :
                    StackShowStructure(testStack);
                    break;
                case 'Q' : case 'q' : // Quit program
                    break;
                default :
                    printf("Wrong command! Retry!\n");
            }
        }
    } while ( cmd!='Q' && cmd!='q' );
    return 0;
}

```