1. 소스 코드

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// 차후에 스택이 필요하면 여기만 복사하여 붙인다.

// ===== 스택 코드의 시작 =====

#define MAX\_STACK\_SIZE 10

typedef int element;

typedef struct {

element data[MAX\_STACK\_SIZE];

int top;

} StackType;

// 스택 초기화 함수

void init\_stack(StackType\* s)

{

s->top = -1;

}

// 공백 상태 검출 함수

int is\_empty(StackType\* s)

{

return (s->top == -1);

}

// 포화 상태 검출 함수

int is\_full(StackType\* s)

{

return (s->top == (MAX\_STACK\_SIZE - 1));

}

// 삽입함수

void push(StackType\* s, element item)

{

if (is\_full(s)) {

fprintf(stderr, "Stack Full\n");

return;

}

else s->data[++(s->top)] = item;

}

// 삭제함수

element pop(StackType\* s)

{

if (is\_empty(s)) {

printf("pop -1\n");

fprintf(stderr, "Stack Empty!\n");

return;

}

else {

printf("pop : %d\n", s->data[(s->top)]);

return s->data[(s->top)--];

}

}

// 피크함수

element peek(StackType\* s)

{

if (is\_empty(s)) {

fprintf(stderr, "스택 공백 에러\n");

exit(1);

}

else return s->data[s->top];

}

// ===== 스택 코드의 끝 =====

int main(void)

{

StackType s;

init\_stack(&s);

srand(time(NULL));

printf("-------using Struct-------\n");

int rand\_num;

for (int i = 0; i < 30; i++)

{

printf("[%d]\t", i);

rand\_num = rand() % 100 + 1;

if (rand\_num % 2 == 0)

{

printf("push : %d\n", rand\_num);

push(&s, rand\_num);

}

else {

pop(&s);

}

}

}

2. 실행화면

텍스트이(가) 표시된 사진

자동 생성된 설명

3. 느낀점

기존의 배열을 사용하여 만들었을 때 보다 구조체를 사용하니 조금 귀찮았지만 만들고 나니 보기 더 편하였다. 또한 ->를 이용하니 더 직관적이어서 좋았다.또한 나중에 element가 바뀌면 element만 바꿔주면 되어서 편할 것 같다는 생각이 들었다.