1. 코드 분석 및 주석

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

typedef int element;

typedef struct DlistNode {//doubly link이니까 llink, rlink 둘다 포함

element data;

struct DlistNode \*llink;

struct DlistNode \*rlink;

} DlistNode;

typedef struct {//top의 주소를 저장하는 구조체 선언

DlistNode \*top;

} LinkedStackType;

void init(LinkedStackType \*s)//초기화 하는 함수

{

s->top = NULL;

}

int is\_empty(LinkedStackType \*s)//top이 null이면 1을 반환하는 empty 확인 함수

{

return (s->top == NULL);

}

void push(LinkedStackType \*s, element item)

{

DlistNode \*temp = (DlistNode \*)malloc(sizeof(DlistNode));

if (temp == NULL) {

fprintf(stderr, "Memory allocation error\n");

return;

}

else {

if (s->top == NULL) {//처음 들어가는 노드라면

temp->data = item;//data에 item 저장하고

s->top = temp;//top 주소를 새로 만든 temp 주소로 저장

}

else {//2번째 이후 노드라면

temp->data = item;

s->top->rlink = temp;//기존의 top 노드의 rlink에 temp를 저장

temp->llink = s->top;//temp의 llink에 기존의 top 주소를 저장

s->top = temp;//이제 top은 새로운 temp를 가르키도록 함

}

}

}

element pop(LinkedStackType \*s)

{

if (is\_empty(s)) {

fprintf(stderr, "Stack is empty\n");

exit(1);

}

else {

DlistNode \*temp = s->top;//일단 temp 주소를 선언해서 top주소를 저장하고

int item = temp->data;//item도 반환하기 위해서 저장해 둔다.

s->top = s->top->llink;//그리고 이제 top은 그 전의 노드를 가르키도록 한다.

free(temp);

return item;

}

}

element peek(LinkedStackType \*s)

{

if (is\_empty(s)) {

fprintf(stderr, "Stack is empty\n");

exit(1);

}

else {

return s->top->data;

}

}

void main()

{

LinkedStackType s;

init(&s);

push(&s, 1);

push(&s, 2);

push(&s, 3);

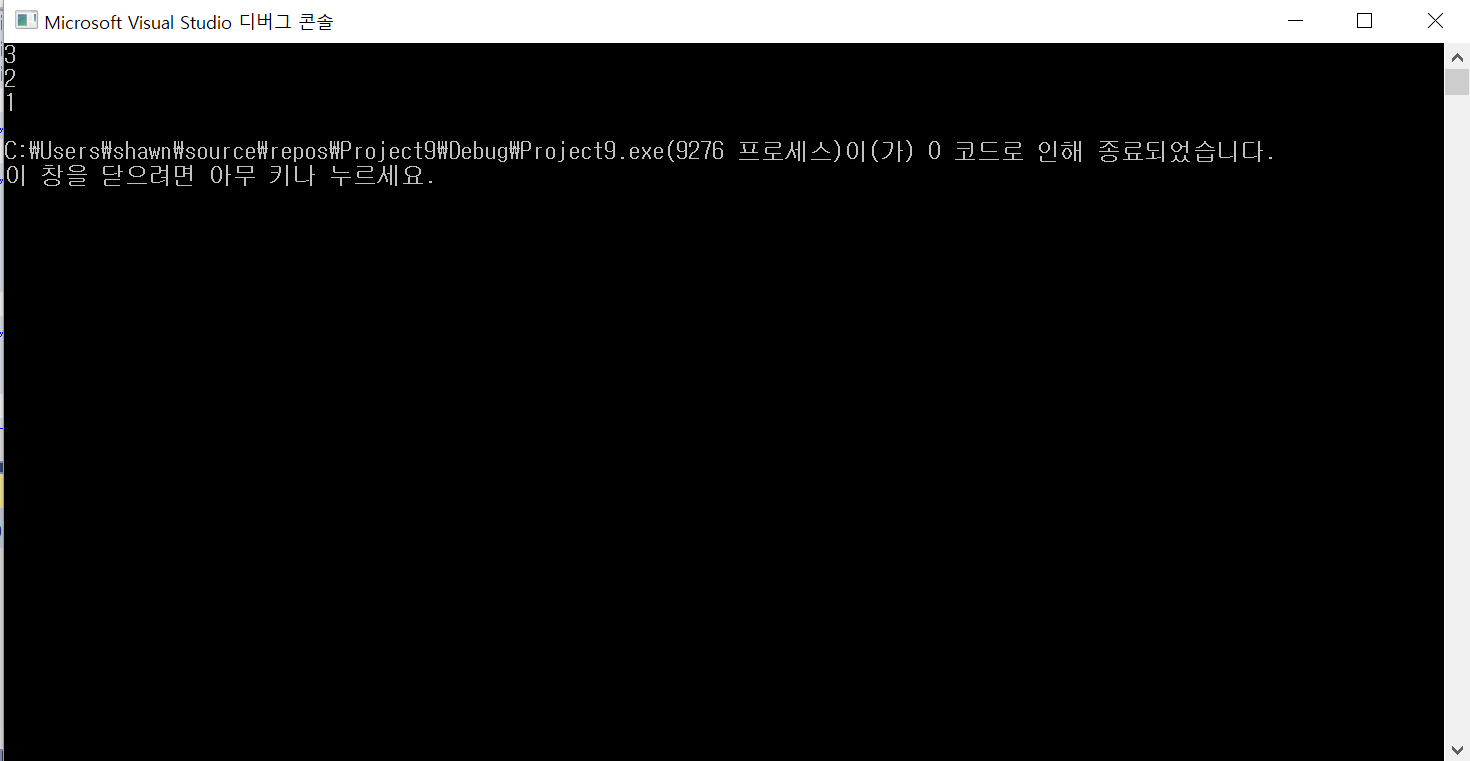
printf("%d\n", pop(&s));

printf("%d\n", pop(&s));

printf("%d\n", pop(&s));

}

2. 결과 및 콘솔 창



3. 코드 분석 및 이유 설명

1) doubly link node로 구성된 스택을 위해 선언한 구조체

typedef struct DlistNode {//doubly link이니까 llink, rlink 둘다 포함

element data;

struct DlistNode \*llink;

struct DlistNode \*rlink;

} DlistNode;

2) doubly linked node의 특징을 활용한 push 함수

else {

if (s->top == NULL) {//처음 들어가는 노드라면

temp->data = item;//data에 item 저장하고

s->top = temp;//top 주소를 새로 만든 temp 주소로 저장

}

else {//2번째 이후 노드라면

temp->data = item;

s->top->rlink = temp;//기존의 top 노드의 rlink에 temp를 저장

temp->llink = s->top;//temp의 llink에 기존의 top 주소를 저장

s->top = temp;//이제 top은 새로운 temp를 가르키도록 함

}

}

3) doubly linked node의 특징을 활용한 pop 함수(llink를 사용해 전에 있는 노드에 접근한다)

else {

DlistNode \*temp = s->top;//일단 temp 주소를 선언해서 top주소를 저장하고

int item = temp->data;//item도 반환하기 위해서 저장해 둔다.

s->top = s->top->llink;//그리고 이제 top은 그 전의 노드를 가르키도록 한다.

free(temp);

return item;

}