#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 100 // 定义栈的最大容量

typedef int data\_t; // 定义栈中元素的数据类型

typedef struct {

data\_t data[N];

int top;

} sqstack;

sqstack \*sqstack\_create() {

sqstack \*l = (sqstack\*)malloc(sizeof(sqstack));

if (l == NULL) {

printf("Memory allocation failed\n");

return NULL;

}

l->top = -1;

return l;

}

int stack\_clear(sqstack \*l) {

if (l == NULL) {

printf("Invalid stack\n");

return -1;

}

l->top = -1;

return 1;

}

int stack\_push(sqstack \*l, data\_t value) {

if (l == NULL) {

printf("Invalid stack\n");

return -1;

}

if (l->top == N - 1) {

printf("Stack is full\n");

return -1;

}

l->top++;

l->data[l->top] = value;

return 1;

}

data\_t stack\_pop(sqstack \*l) {

if (l == NULL || l->top == -1) {

printf("Stack is empty\n");

return -1;

}

data\_t popped\_value = l->data[l->top];

l->top--;

return popped\_value;

}

int stack\_empty(sqstack \*l) {

if (l == NULL) {

printf("Invalid stack\n");

return -1;

}

return (l->top == -1 ? 1 : 0);

}

int stack\_full(sqstack \*l) {

if (l == NULL) {

printf("Invalid stack\n");

return -1;

}

return (l->top == N - 1 ? 1 : 0);

}

void stack\_free(sqstack \*l) {

if (l != NULL) {

free(l);

}

}

void stack\_show(sqstack \*l) {

if (l == NULL) {

printf("Invalid stack\n");

return;

}

for (int i = 0; i <= l->top; i++) {

printf("%d ", l->data[i]);

}

printf("\n");

}

int main() {

sqstack \*stack = sqstack\_create();

// 入栈操作

stack\_push(stack, 10);

stack\_push(stack, 20);

stack\_push(stack, 30);

// 出栈操作

data\_t popped\_value = stack\_pop(stack);

printf("Popped value: %d\n", popped\_value);

// 判断栈是否为空

if (stack\_empty(stack)) {

printf("Stack is empty\n");

} else {

printf("Stack is not empty\n");

}

// 判断栈是否已满

if (stack\_full(stack)) {

printf("Stack is full\n");

} else {

printf("Stack is not full\n");

}

// 展示栈中元素

printf("Stack elements: ");

stack\_show(stack);

// 清空栈

stack\_clear(stack);

// 释放栈内存

stack\_free(stack);

return 0;

}