#include <stdio.h>

#include <stdlib.h>

// Define a maximum size for the static stack

#define MAX\_SIZE 100

// Define a structure for the stack

struct Stack {

int data[MAX\_SIZE];

int top;

};

// Function to initialize the stack

void initialize(struct Stack \*stack) {

stack->top = -1;

}

// Function to push an element onto the stack

int push(struct Stack \*stack, int value) {

if (stack->top >= MAX\_SIZE - 1) {

printf("Stack Overflow\n");

return 0; // Return 0 to indicate failure

}

stack->data[++stack->top] = value;

return 1; // Return 1 to indicate success

}

// Function to pop an element from the stack

int pop(struct Stack \*stack, int \*value) {

if (stack->top < 0) {

printf("Stack Underflow\n");

return 0; // Return 0 to indicate failure

}

\*value = stack->data[stack->top--];

return 1; // Return 1 to indicate success

}

// Function to check if the stack is empty

int isEmpty(struct Stack \*stack) {

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

}

// Function to check if the stack is full (only for static implementation)

int isFull(struct Stack \*stack) {

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

}

int main() {

// Static stack

struct Stack staticStack;

initialize(&staticStack);

printf("Static Stack Implementation:\n");

for (int i = 1; i <= 5; i++) {

if (push(&staticStack, i))

printf("Pushed %d onto the stack\n", i);

}

while (!isEmpty(&staticStack)) {

int value;

if (pop(&staticStack, &value))

printf("Popped %d from the stack\n", value);

}

// Dynamic stack

struct Stack \*dynamicStack = (struct Stack \*)malloc(sizeof(struct Stack));

initialize(dynamicStack);

printf("\nDynamic Stack Implementation:\n");

for (int i = 1; i <= 5; i++) {

if (push(dynamicStack, i))

printf("Pushed %d onto the stack\n", i);

}

while (!isEmpty(dynamicStack)) {

int value;

if (pop(dynamicStack, &value))

printf("Popped %d from the stack\n", value);

}

free(dynamicStack); // Don't forget to free the dynamically allocated memory

return 0;

}

**A screenshot of a computer

Description automatically generated**