program to convert of postfix expression using stack in c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_SIZE 100

// Stack data structure

struct Stack {

int top;

char items[MAX\_SIZE];

};

// Push an item to the stack

void push(struct Stack\* stack, char item) {

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

printf("Stack Overflow\n");

}

else {

stack->top++;

stack->items[stack->top] = item;

}

}

// Pop an item from the stack

char pop(struct Stack\* stack) {

if (stack->top == -1) {

printf("Stack Underflow\n");

return '\0';

}

else {

char item = stack->items[stack->top];

stack->top--;

return item;

}

}

// Convert postfix expression to infix expression

void postfixToInfix(char\* postfix, char\* infix) {

int i;

char item1, item2, symbol;

struct Stack stack;

// Initialize stack

stack.top = -1;

// Traverse postfix expression

for (i = 0; postfix[i] != '\0'; i++) {

// If current character is an operand

if (isalnum(postfix[i])) {

push(&stack, postfix[i]);

}

// If current character is an operator

else if (postfix[i] == '+' || postfix[i] == '-' || postfix[i] == '\*' || postfix[i] == '/') {

item1 = pop(&stack);

item2 = pop(&stack);

symbol = postfix[i];

sprintf(infix, "(%c%c%c)", item2, symbol, item1);

push(&stack, \*infix);

}

}

}

// Main function

int main() {

char postfix[MAX\_SIZE], infix[MAX\_SIZE];

printf("Enter postfix expression: ");

scanf("%s", postfix);

postfixToInfix(postfix, infix);

printf("Infix expression: %s\n", infix);

return 0;

}

