program to evaluate prefix expression using stack in c

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX\_SIZE 100

// Stack data structure

struct Stack {

int top;

int items[MAX\_SIZE];

};

// Push an item to the stack

void push(struct Stack\* stack, int 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

int pop(struct Stack\* stack) {

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

printf("Stack Underflow\n");

return -1;

}

else {

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

stack->top--;

return item;

}

}

// Evaluate prefix expression

int evaluatePrefix(char\* prefix) {

int i, operand1, operand2, result, len;

struct Stack stack;

// Initialize stack

stack.top = -1;

// Get length of prefix expression

len = strlen(prefix);

// Traverse prefix expression in reverse order

for (i = len - 1; i >= 0; i--) {

// If current character is an operand

if (isdigit(prefix[i])) {

push(&stack, prefix[i] - '0');

}

// If current character is an operator

else {

operand1 = pop(&stack);

operand2 = pop(&stack);

switch (prefix[i]) {

case '+':

result = operand1 + operand2;

break;

case '-':

result = operand1 - operand2;

break;

case '\*':

result = operand1 \* operand2;

break;

case '/':

result = operand1 / operand2;

break;

default:

printf("Invalid operator\n");

return -1;

}

push(&stack, result);

}

}

return pop(&stack);

}

// Main function

int main() {

char prefix[MAX\_SIZE];

printf("Enter prefix expression: ");

scanf("%s", prefix);

int result = evaluatePrefix(prefix);

printf("Result: %d\n", result);

return 0;

}

