









/\* Author: Malachy Crossan

\* Date: 10-14-24

\* Class: EGN3211-0002

\* Assignment: 4

\* Prompts user for postfix expression

\* Then calculates the result

\* Optionally outputs the stack at each iteration

\* \*/

#include <stdio.h>

#include <math.h>

#define MAX\_EXPRESSION\_LENGTH 200

double add (double a, double b); // adds a to b

double sub (double a, double b); // subtracts b from a

double mut (double a, double b); // multiplies a by b

double dvd (double a, double b); // divides a by b

// double pow(double,double) math.h returns a^b

// double exp(double) math.h returns e^a

double fac (double a); // returns a!

int parseType (char expression); // +:0 -:1 \*:2 /:3 ^:4 e:5 !:6 0->9:7 ' ':8

int main (void) {

double (\*twoInOp[5])(double, double) = {add, sub, mut, dvd, pow}; //reference functions with two inputs

double (\*oneInOp[2])(double) = {exp, fac}; // reference functions with one input

// Prompt user

printf("Input an expression to be solved: ");

char expression[MAX\_EXPRESSION\_LENGTH];

scanf("%[^\n]", &expression);

// initialize stack

int top = -1;

double stack[MAX\_EXPRESSION\_LENGTH/2];

// "curr" pointer loops through the expression string

char \*curr = expression;

while (\*curr != '\0') {

int type = parseType(\*curr);

if (type < 0) {curr++; continue;} // unknown characters are ignored

else if (type <= 4)

stack[++top] = (\*twoInOp[type])(stack[top--],stack[top--]); //two input operation pops inputs off stack then pushes result

else if (type <= 6)

stack[++top] = (\*oneInOp[type-5])(stack[top--]); //one input operation pops input off stack then pushes result

else if (type == 7) {

double val = 0;

// loops through expression until non-digit character is found

while (parseType(\*curr) == 7) {

val \*= 10; // shifts decimal place to the right

val += \*curr - '0'; // inserts one's place digit (ascii for integer n minus ascill 0 equals integer n)

curr++;

}

stack[++top] = val; // Pushes final value onto stack

} else { curr++; continue; }

for (int i = 0; i <= top; i++) printf("[%.1f]",stack[i]); printf("<-stack %c\n",\*curr); //prints the current stack

curr++;

}

printf("Result: %lf\n",stack[top]);

}

double add (double a, double b) {return a + b;};

double sub (double a, double b) {return a - b;};

double mut (double a, double b) {return a \* b;};

double dvd (double a, double b) {return a / b;};

double etx (double a) {return exp(a);}

double fac (double a) {return a \* tgamma(a);} // a! = a \* gamma(a) for all real numbers

int parseType (char expression) {

switch (expression) {

case '+':

return 0;

case '-':

return 1;

case '\*':

return 2;

case '/':

return 3;

case '^':

return 4;

case 'e':

return 5;

case '!':

return 6;

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

return 7;

case ' ':

return 8;

}

return -1;

}