#include <stdio.h>

#include <stdlib.h>

#include <string.h>

enum TokenType {

INTEGER,

PLUS,

MINUS,

MULTIPLY,

DIVIDE,

LPAREN,

RPAREN,

END

};

typedef struct {

enum TokenType type;

int value;

} Token;

Token tokens[100];

int token\_count = 0;

int current\_token = 0;

void tokenize(char\* input) {

token\_count = 0;

int i = 0;

while (input[i] != '\0') {

while (input[i] == ' ' || input[i] == '\t') i++;

if (input[i] >= '0' && input[i] <= '9') {

int value = 0;

while (input[i] >= '0' && input[i] <= '9') {

value = value \* 10 + (input[i] - '0');

i++;

}

tokens[token\_count].type = INTEGER;

tokens[token\_count].value = value;

token\_count++;

continue;

}

switch (input[i]) {

case '+':

tokens[token\_count].type = PLUS;

break;

case '-':

tokens[token\_count].type = MINUS;

break;

case '\*':

tokens[token\_count].type = MULTIPLY;

break;

case '/':

tokens[token\_count].type = DIVIDE;

break;

case '(':

tokens[token\_count].type = LPAREN;

break;

case ')':

tokens[token\_count].type = RPAREN;

break;

default:

printf("Unexpected character: %c\n", input[i]);

return;

}

token\_count++;

i++;

}

tokens[token\_count].type = END;

}

int parse\_expression();

int parse\_factor() {

Token token = tokens[current\_token];

if (token.type == INTEGER) {

current\_token++;

return token.value;

}

if (token.type == LPAREN) {

current\_token++;

int result = parse\_expression();

if (tokens[current\_token].type != RPAREN) {

printf("Expected closing parenthesis\n");

return 0;

}

current\_token++;

return result;

}

if (token.type == MINUS) {

current\_token++;

return -parse\_factor();

}

printf("Unexpected token in factor\n");

return 0;

}

int parse\_term() {

int left = parse\_factor();

while (1) {

Token token = tokens[current\_token];

if (token.type == MULTIPLY) {

current\_token++;

int right = parse\_factor();

left \*= right;

}

else if (token.type == DIVIDE) {

current\_token++;

int right = parse\_factor();

if (right == 0) {

printf("Division by zero\n");

return 0;

}

left /= right;

}

else {

break;

}

}

return left;

}

int parse\_expression() {

int left = parse\_term();

while (1) {

Token token = tokens[current\_token];

if (token.type == PLUS) {

current\_token++;

int right = parse\_term();

left += right;

}

else if (token.type == MINUS) {

current\_token++;

int right = parse\_term();

left -= right;

}

else {

break;

}

}

return left;

}

int main() {

char input[100];

printf("Simple Calculator\n");

printf("Enter an expression (e.g., 5+3\*2): ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = 0;

tokenize(input);

current\_token = 0;

int result = parse\_expression();

if (tokens[current\_token].type == END) {

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

} else {

printf("Parsing error: unexpected tokens at end\n");

}

return 0;

}