//Samuel Larson

//Project 6

//10/22/18

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class stack

{

public:

typedef char item;

//member constant

static const int CAPACITY = 20;

//constructor

stack(){used = 0;}

//modification functions

void push(item entry);

item pop();

//constant member functions

int size(){return used;}

bool empty(){return used == 0;}

private:

//data members

item data[CAPACITY];

int used;

};

int main()

{

stack s;

ifstream infile;

infile.open("equation.txt");

char ch;

string line;

int result, oper1, oper2, pos;

char func;

while(getline(infile, line))//reads in a line from the file while one is available

{

cout << "expression: ";

pos = 0;

while(line[pos] >= ' ')//reads a character from the line while it is printable

{

ch = line[pos];

cout << ch;

pos++;

s.push(ch);

if(ch < '\*' || ch > '9' || ch == ',' || ch == '.')//makes sure the character is valid

{

cout << endl << "expression not valid" << endl

<< "ending program" << endl;

return 0;

}

if(ch == '+' || ch == '-' || ch == '\*' || ch == '/') //checks if the character is an operator

{

func = s.pop();//gets the operator from the stack

if(s.empty())//makes sure the stack isn’t empty

{

cout << endl << "expression not valid" << endl

<< "ending program" << endl;

return 0;

}

oper2 = s.pop() - '0';//gets the 2nd operand from the stack

if(s.empty())//makes sure the stack isn’t empty

{

cout << endl << "expression not valid" << endl

<< "ending program" << endl;

return 0;

}

oper1 = s.pop() - '0';//gets the 1st operand

switch(func)//performs operation

{

case'+':

result = oper1 + oper2;

ch = '0' + result;

s.push(ch);

break;

case'-':

result = oper1 - oper2;

ch = '0' + result;

s.push(ch);

break;

case'\*':

result = oper1 \* oper2;

ch = '0' + result;

s.push(ch);

break;

case'/':

result = oper1 / oper2;

ch = '0' + result;

s.push(ch);

break;

}

}

}

cout << endl << "Value = " << s.pop() << endl;//prints results

}

return 0;

}

//precondition needs a character to push postcondition character is added to the stack

void stack::push(item entry)

{

data[used] = entry;

++used;

}

//precondition a stack with something in it postcondition a character is returned

stack:: item stack::pop()

{

--used;

return data[used];

}