#include <iostream>

#include <cstring>

#include "rpn.h"

using namespace std;

string a = "",b = "",c = "",d = "";

/\* if any cell in the matrix is equation, it will be solved \*/

string decrypteMatrix(string matrix){

string decryptedMatrix = "";

string tmp = "";

for(int i = 0 ; i < matrix.length(); i ++){

if(matrix[i] != ';' && matrix[i] != separator){

tmp += matrix[i];

}else{

if(decryptedMatrix != "" && decryptedMatrix[decryptedMatrix.length()-1] != ';'){

decryptedMatrix += separator;

}

decryptedMatrix += solveEquation(tmp);

tmp = "";

if(matrix[i] == ';'){

decryptedMatrix += ';';

}

}//end else

}//end for

if(decryptedMatrix != ""){

decryptedMatrix += separator;

}

decryptedMatrix += solveEquation(tmp);

return decryptedMatrix;

}

/\* prints the matrix of screen \*/

void showMatrix(string matrix){

char tmp;

cout<< "\n";

for(int i = 0 ; i < matrix.length() ; i ++){

tmp = matrix[i];

if(tmp != separator && tmp != ';'){

cout<< matrix[i];

}else if(tmp == separator){

cout<< "\t";

}else if(tmp == ';'){

cout<< "\n";

}

}

cout<< "\n";

}

/\* returns the number of columns in the matrix \*/

int columnNum(string matrix){

int n = 0;

for(int i = 0 ; i < matrix.length() ; i ++){

if(matrix[i] == ';'){

break;

}else if(matrix[i] == separator){

n++;

}

}

return ++n;

}

/\* returns the number of rows in the matrix \*/

int rowNum(string matrix){

int m = 0;

for(int i = 0 ; i < matrix.length() ; i ++){

if(matrix[i] == ';'){

m++;

}

}

return ++m;

}

/\* gets the matrix from the user \*/

void enterMatrix(){

cout<< "enter matrix ex: (a=159.15 15.3 12;8 9+8\*(10/5)-3 8-5) : \n";

string matrix = "";

char tmp;

int m = 0,n = 0;

getline(cin,matrix);

tmp = matrix[0];

matrix.erase(0,2);

matrix = decrypteMatrix(matrix);

showMatrix(matrix);

/\* assigning the data to the matrix \*/

switch(tmp){

case 'a':

a = matrix;

break;

case 'b':

b = matrix;

break;

case 'c':

c = matrix;

break;

case 'd':

d = matrix;

break;

}

}

/\* gets the row with number m in the matrix \*/

string Row(int m,string matrix)

{

int semicolons = 0;

string tmp = "";

if( m > rowNum(matrix) )

{

cout<< "doesn't exist\n";

return "";

}

for(int i = 0 ; i < matrix.length() ; i ++){

if(matrix[i] != ';'){

tmp += matrix[i];

}else{

semicolons++;

if(semicolons == m){

return tmp;

}

tmp = "";

}//end else

}//end for

return tmp;

}

/\* returns the cell with row m and column n in the matrix \*/

string cell(int m,int n,string matrix){

return token(n,Row(m,matrix));

}

/\* matrix1 + matrix2 \*/

string addition(string v1,string v2){

string result = "";

double d1 = 0.0, d2 = 0.0;

int m,n;

if(rowNum(v1) != rowNum(v2) || columnNum(v1) != columnNum(v2)){

cout<< "can't perform this equation\n";

return "";

}

m = rowNum(v1);

n = columnNum(v1);

for(int i = 0 ; i < m ; i ++){

for(int j = 0 ; j < n ; j++){

d1 = string\_To\_Double(cell(i+1,j+1,v1));

d2 = string\_To\_Double(cell(i+1,j+1,v2));

result += double\_To\_String(d1+d2);

if( j != n-1){

result += separator;

}else{

result += ';';

}

}//end nested for

}//end for

return result;

}

/\* matrix1 - matrix2 \*/

string substraction(string v1,string v2){

string result = "";

double d1 = 0.0, d2 = 0.0;

int m,n;

if(rowNum(v1) != rowNum(v2) || columnNum(v1) != columnNum(v2)){

cout<< "can't perform this equation\n";

return "";

}

m = rowNum(v1);

n = columnNum(v1);

for(int i = 0 ; i < m ; i ++){

for(int j = 0 ; j < n ; j++){

d1 = string\_To\_Double(cell(i+1,j+1,v1));

d2 = string\_To\_Double(cell(i+1,j+1,v2));

result += double\_To\_String(d1-d2);

if( j != n-1){

result += separator;

}else{

result += ';';

}

}//end nested for

}//end for

return result;

}

/\* matrix1 \* matrix2 \*/

string multiplication(string v1,string v2){

string result = "";

double tmp;

double d1,d2;

if(columnNum(v1) != rowNum(v2)){

cout<< "can't perform this equation\n";

return "";

}

for(int m = 0 ; m < rowNum(v1) ; m ++){

for(int n = 0 ; n < columnNum(v2) ; n ++){

tmp = 0.0;

for(int i = 0 ; i < columnNum(v1)/\* same as rowNum(v2) \*/ ; i ++){

d1 = string\_To\_Double(cell(m+1,i+1,v1));

d2 = string\_To\_Double(cell(i+1,n+1,v2));

tmp += d1\*d2;

}

result += double\_To\_String(tmp);

if(n != columnNum(v2)-1){

if(result != ""){

result += separator;

}

}else{

result += ';';

}

}

}

return result;

}

/\* matrix1 / matrix2 \*/

string division(string v1,string v2){

string result = "";

double tmp;

double d1,d2;

if(columnNum(v1) != rowNum(v2)){

cout<< "can't perform this equation\n";

return "";

}

for(int m = 0 ; m < rowNum(v1) ; m ++){

for(int n = 0 ; n < columnNum(v2) ; n ++){

tmp = 0.0;

for(int i = 0 ; i < columnNum(v1)/\* same as rowNum(v2) \*/ ; i ++){

d1 = string\_To\_Double(cell(m+1,i+1,v1));

d2 = string\_To\_Double(cell(i+1,n+1,v2));

tmp += d1/d2;

}

result += double\_To\_String(tmp);

if(n != columnNum(v2)-1){

if(result != ""){

result += separator;

}

}else{

result += ';';

}

}

}

return result;

}

/\* understands the equation and solve it \*/

void operateMatrices(){

string equation = "";

string result = "";

string v1,v2;

cout<< "enter your equation ex: ( c=a+b ) : ";

getline(cin,equation);

switch(equation[2]){

case 'a':

v1 = a;

break;

case 'b':

v1 = b;

break;

case 'c':

v1 = c;

break;

case 'd':

v1 = d;

break;

}

switch(equation[4]){

case 'a':

v2 = a;

break;

case 'b':

v2 = b;

break;

case 'c':

v2 = c;

break;

case 'd':

v2 = d;

break;

}

switch(equation[3]){

case '+':

result = addition(v1,v2);

break;

case '-':

result = substraction(v1,v2);

break;

case '\*':

result = multiplication(v1,v2);

break;

case '/':

result = division(v1,v2);

break;

}

switch(equation[0]){

case 'a':

a = result;

break;

case 'b':

b = result;

break;

case 'c':

c = result;

break;

case 'd':

d = result;

break;

}

}

void main(){

while(true){

cout<< "\n1 to enter data, 2 to operate on matrices, 3 to show all : ";

string tmp = "";

getline(cin,tmp);

cout<< "\n";

if(tmp.compare("1") == 0){

enterMatrix();

}else if(tmp.compare("2") == 0){

operateMatrices();

}else if(tmp.compare("3") == 0){

//cout<< "\n\t\ta\n";

showMatrix(a);

//cout<< "\n\t\tb\n";

showMatrix(b);

//cout<< "\n\t\tc\n";

showMatrix(c);

//cout<< "\n\t\td\n";

showMatrix(d);

}else{

exit(0);

}

}//end while

}