Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ОТЧЕТ

по лабораторной работе №1

Тема работы: Действия над матрицами

Выполнил

студент: гр. 251003 Панкратьев Е.С.

Проверил: Фадеева Е.П.

Минск 2023

Содержание

[1 Постановка задачи 3](#_Toc129101132)

[2 Метод решения 4](#_Toc129101133)

[2.1 Постановка задачи 4](#_Toc129101134)

[2.2 Сложение матриц 4](#_Toc129101135)

[2.3 Вычитание матриц 5](#_Toc129101136)

[2.4 Умножение матрицы на число 5](#_Toc129101137)

[2.5 Умножение матриц 5](#_Toc129101138)

[2.6 Возведение матрицы в квадрат 6](#_Toc129101139)

[2.7 Описание оператора try..except..end 6](#_Toc129101140)

[2.8 Проверка введенных данных 6](#_Toc129101141)

[3 Описание алгоритмов решения задачи 7](#_Toc129101142)

[4 Структура данных 9](#_Toc129101143)

[5 Схема алгоритма решения задачи по ГОСТ 19.701-90 11](#_Toc129101144)

[5.1 Схема алгоритма MatrixSum 12](#_Toc129101145)

[5.2 Схема алгоритма MultiplyMatrixAndConst 13](#_Toc129101146)

[5.3 Схема алгоритма MultiplyMatrices 14](#_Toc129101147)

[5.4 Схема алгоритма InputMatrix 15](#_Toc129101148)

[5.5 Cхема алгоритма OutputMatrix 16](#_Toc129101149)

[6 Результаты расчетов 17](#_Toc129101150)

[Приложение А 18](#_Toc129101151)

[Приложение Б 22](#_Toc129101152)

[Приложение В 24](#_Toc129101153)

[Приложение Г 28](#_Toc129101154)

# Постановка задачи

Разработать программу, которая вычисляет следующее матричное выражение:

Матрицы *A* и *В* имеют порядок 3. Пользователь заполняет матрицы *A* и *В*.

Необходимо вывести промежуточные и окончательные результаты расчетов.

# Метод решения

## Постановка задачи

Необходимо разработать подпрограммы, которые будут выполнять операции над матрицами, которые присутствуют в матричном выражении. Также необходимо разработать подпрограммы, которые выполняют ввод и вывод матриц.

В матричном выражении присутствуют следующие операции:

* сумма матриц;
* разность матриц;
* умножение матрицы на число;
* умножение матриц;
* возведение матрицы в квадрат.

Далее рассмотрим каждую операцию подробнее.

## Сложение матриц

**Сложение матриц***А* и *Б* – это арифметическая операция, в результате которой, должна получаться матрица *С*, каждый элемент которой равен сумме соответствующих элементов складываемых матриц:

Более подробно формула сложения двух матриц выглядит так:

Для расчета суммы используется два цикла, обеспечивая проход по всем значениям матриц.

## Вычитание матриц

**Вычитание матриц** полностью аналогично по алгоритму сложения, только знак минус. Каждый элемент искомой матрицы C получается благодаря вычитанию соответствующих элементов матриц A и B:

Запишем подробную формулу вычитания двух матриц:

## Умножение матрицы на число

**Умножение матрицы на число** - это операция над матрицей, в результате которой каждый её элемент умножается на действительное число:

Стоит заметить, что получаемая матрица *С* в результате имеет тот же размер, что и матрица *A*.

Для расчета умножения матрицы на число также используется два цикла, чтобы обеспечить проход по всем значениям матрицы.

## Умножение матриц

Операция умножения матрицы *А* на матрицу *В* определена только для согласованных матриц, у которых число столбцов матрицы *А* равно числу строк матрицы *В*:

Для того, чтобы найти произведение матриц нужно строки левой матрицы умножить на столбцы правой матрицы. Умножение производим по правилу скалярного произведения. То есть находим сумму произведений соответствующих элементов. Например, при умножении первой строки на первый столбец получаем

Для расчета произведения матриц используется три цикла. Два цикла определяют номер строки и столбца результирующей матрицы *С*, а третий обеспечивает проход по значениям строк и столбцов матриц *А* и *В*.

## Возведение матрицы в квадрат

Чтобы возвести матрицу *A* в квадрат, нужно это матрицу умножить саму на себя:

Умножение матриц [см. главу 2.5 Умножение матриц](#_Умножение_матриц)

## Описание оператора try..except..end

Оператор try..except..end имеет вид:

try  
 операторы;  
except  
  блок обработки исключений;  
end;

Выполнение блока начинается с секции try, при отсутствии исключительных ситуаций только она и выполняется. Секция except получает управление в случае возникновения исключительной ситуации. После обработки выполняются операторы, стоящие после end.

## Проверка введенных данных

Элементы матриц должны быть целочисленные числа. Проверка введенных данных происходит с помощью цикла с предусловием repeat..until, чтобы при вводе некорректных данных пользователь заново заполнял их. В теле цикла с помощью оператора try..except..end (описание оператора [см. главу 2.7](#_Описание_оператора_try..except..end)) проверяем соответствие с типом данным.

# Описание алгоритмов решения задачи

Таблица – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Название алгоритма | Формальные  параметры | Предпола-гаемый тип реализации |
| 1 | Основной алгоритм | Вызов следующих подпрограмм:  MatrixSum, MultiplyMatrixAndConst, MultiplyMatrices, InputMatrix,  OutputMatrix. |  |  |
| 2 | SumMat  (  FirstMatrix,  SecondMatrix,  isDifference,  Result  ) | Выполняет сложение матрицы FirstMatrix с матрицей SecondMatrix. Сохраняет результать расчета в матрице Result.  isDifference указывает на выполняемое действие: если значение isDifference истинно, то следует выполнить вычитание, в противном случае следует выполнить сложение. | FirstMatrix – получает от фактического параметра адрес c защитой;  SecondMatrix – получает от фактического параметра адрес c защитой;  isDifference – получает от фактического параметра адрес с защитой;  Result  – получает от фактического параметра адрес, возвращаемый параметр | Процедура |
| 3 | MultiplyMatrixAndConst  (  Matrix,  Num,  Result  ) | Умножает матрицу Matrix на константу Num c сохранением результата в Result | Matrix – получает от фактического параметра адрес c защитой;  Num – получает от фактического параметра адрес c защитой; | Процедура |

Продолжение Таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Result  – получает от фактического параметра адрес,  возвращаемый параметр |  |
| 4 | MultiplyMatrices  (  FirstMatrix,  SecondMatrix,  Result  ) | Умножает матрицу FirstMatrix на матрицу SecondMatrix c сохранением результатов в матрице Result | FirstMatrix – получает от фактического параметра адрес c защитой;  SecondMatrix – получает от фактического параметра адрес c защитой;  Result – получает от фактического параметра адрес, возвращаемый параметр | Процедура |
| 5 | InputMatrix  (  Result  ) | Ввод матрицы. Результат сохраняется в матрицу Result |  | Процедура |
| 6 | OuputMat  (  Matrix  ) | Вывод матрицы Matrix | Matrix –получает от фактического параметра адрес c защитой | Процедура |

# Структура данных

Таблица – Структура данных основной программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| LeftExpressionPart, LeftExpression, RightExpressionPart, RightExpression | Array [1..3, 1..3] of Integer | Массивы, предназначенные для хранения промежуточных расчетов |
| ResultOfExpression | Array [1..3, 1..3] of Integer | Массив, который хранит результат матричного выражения |
| MatrixA, MatrixB | Array [1..3, 1..3] of Integer | Исходные матрицы |

Таблица – Структура данных алгоритма MatrixSum (FirstMatrix, SecondMatrix, isDifference, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| MatrixSum, SecondMatrix | Array [1..3, 1..3] of Integer | Исходные матрицы | Формальный |
| isDifference | Boolean | Переменная для реализации сложения/вычитания | Формальный |
| Result | Array [1..3, 1..3] of Integer | Матрица с результатами расчета | Формальный |

Таблица – Структура данных алгоритма MultiplyMatrixAndConst (Matrix, Num, Result)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | | Тип параметра |
| i, j | Integer | Счетчики циклов | | Локальный |
| Matrix | Array [1..3, 1..3] of Integer | Исходная матрица | Формальный | |
| Num | Integer | Коэффициент умножения | Формальный | |
| Result | Array [1..3, 1..3] of Integer | Матрица с результатами расчета | Формальный | |

Таблица – Структура данных алгоритма MultiplyMatrices (FirstMatrix, SecondMatrix, Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j, l | Integer | Счетчики циклов | Локальный |
| FirstMatrix,  SecondMatrix | Array [1..3, 1..3] of Integer | Исходные матрицы | Формальный |
| Result | Array [1..3, 1..3] of Integer | Матрица с результатами расчета | Формальный |

Таблица 6 – Структура данных алгоритма InputMatrix (Result)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| isCorrect | Boolean | Индикатор правильного ответа | Локальный |
| Result | Array [1..3, 1..3] of Integer | Введенная матрица | Формальный |

Таблица 7 – Структура данных алгоритма OutputMat (Matrix)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| i, j | Integer | Счетчики циклов | Локальный |
| Matrix | Array [1..3, 1..3] of Integer | Матрица, которую необходимо вывести | Формальный |

# Схема алгоритма решения задачи по ГОСТ 19.701-90



Рисунок – Схема алгоритма в соответствии с ГОСТ 19.701-90

## Схема алгоритма MatrixSum



Рисунок 2 – Схема алгоритма MatrixSum

## Схема алгоритма MultiplyMatrixAndConst



Рисунок – Схема алгоритма MultiplyMatrixAndConst

## Схема алгоритма MultiplyMatrices



Рисунок – Схема алгоритма MultiplyMatrices

## Схема алгоритма InputMatrix



Рисунок – Схема алгоритма InputMatrix

## Cхема алгоритма OutputMatrix



Рисунок 6 – Схема алгоритма OutputMatrix

# Результаты расчетов

В ходе работы были получены следующие результаты:

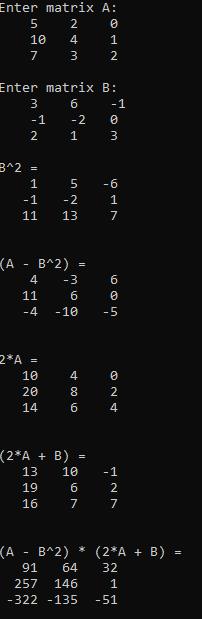


Рисунок 7 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы

Program Lab1FuncImmediately;

{

Matrices A and B have order 3. The user fills in matrices

A and B.

The program finds the value of the matrix expression: (A –

B^2) \* (2\*A + B).

}

{$APPTYPE CONSOLE}

Type

TMatrix = Array[1..3, 1..3] of Integer;

// TMatrix - type for a matrix of order 3

Var

MatrixA: TMatrix;

MatrixB: TMatrix;

LeftExpressionPart, LeftExpression: TMatrix;

RightExpressionPart, RightExpression: TMatrix;

ResultOfExpression: TMatrix;

// FirstMatrix, SecondMatrix - matrix A and B respectively

// ResultOfExpression - the result of the matrix expression

// The function finds the sum (difference) of two matrices.

// For the difference, set the third parameter to true

Procedure MatrixSum(Const FirstMatrix, SecondMatrix:

TMatrix; var Result: TMatrix; Const isDifference:

Boolean = False);

Var

i, j, Sign: Integer;

// i,j – cycle counter

// Sign – sign

Begin

// Determine the sign

If isDifference then

Sign:= -1

Else

Sign:= 1;

// Sum the matrices

For i:= 1 to 3 do

For j:= 1 to 3 do

Result[i,j]:=FirstMatrix[i,j]+Sign\*SecondMatrix[i,j];

End;

// The function finds the product of a matrix and a number

Procedure MultiplyMatrixAndConst(Const Matrix: TMatrix;

Const Num: Integer; var Result: TMatrix);

Var

i, j:integer;

// i, j - cycle counter

Begin

// Finding the product

For i:= 1 to 3 do

For j:= 1 to 3 do

Result[i,j]:=Num\*Matrix[i,j];

End;

// The function finds the product of two matrices (matrices

// must be the same size)

Procedure MultiplyMatrices(Const FirstMatrix, SecondMatrix:

TMatrix; var Result: TMatrix);

Var

i, j, l: Integer;

// i, j, l - cycle counter

Begin

// Multiplying matrices

For i:= 1 to 3 do

For j:= 1 to 3 do

Begin

// Initializing the value of matrix element

Result[i,j]:= 0;

For l:= 1 to 3 do

Result[i,j]:= Result[i,j] + FirstMatrix[i,l] \*

SecondMatrix[l,j];

End;

End;

// Function to read a matrix

Procedure InputMatrix(var Result: TMatrix);

Var

i,j: integer;

isCorrect: boolean;

// i, j - cycle counter

// isCorrect - flag to confirm the correctness of entering

// numbers

Begin

// Matrix input

for i := 1 to 3 do

begin

for j := 1 to 3 do

repeat

isCorrect:= True;

Try

Read(Result[i, j]);

Except

isCorrect:= False;

Writeln('Matrix value entered incorrectly. Repeat

again');

End;

until isCorrect;

Readln;

end;

End;

// The procedure outputs a matrix

Procedure OutputMatrix(Const Matrix: TMatrix);

Var

i,j:integer;

// i, j - cycle counter

Begin

// Matrix output

For i:= 1 to 3 do

Begin

For j:= 1 to 3 do

Write(Matrix[i,j]:5);

Writeln

End;

Writeln

End;

Begin

// Outputting the initial matrixes

Writeln('Enter matrix A:');

InputMatrix(MatrixA);

Writeln;

Writeln('Enter matrix B:');

InputMatrix(MatrixB);

Writeln;

// Calculating the value of the expression

Writeln('(A - B^2) \* (2\*A + B)=');

// (A - B^2)

MultiplyMatrices(MatrixB, MatrixB, LeftExpressionPart);

MatrixSum(MatrixA, LeftExpressionPart, LeftExpression,

True);

// (2\*A + B)

MultiplyMatrixAndConst(MatrixA, 2, RightExpressionPart);

MatrixSum(RightExpressionPart, MatrixB, RightExpression);

// (A - B^2) \* (2\*A + B)

MultiplyMatrices(LeftExpression, RightExpression,

ResultOfExpression);

OutputMatrix(ResultOfExpression);

Readln;

End.

Приложение Б

(обязательное)

Тестовые наборы

**Тестовая ситуация: некорректный ввод данных**

Исходные данные: Некорректный ввод системый счисления

Ожидаемый результат: Повторная попытка

Полученный результат:

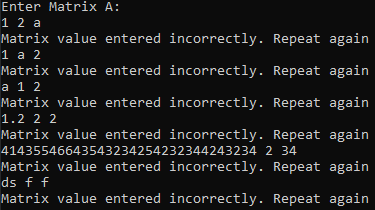


Рисунок 8 – Результаты расчетов

**Тестовая ситуация: корректный ввод данных**

Таблица – Тестовые наборы

|  |  |  |
| --- | --- | --- |
| Тест | Исходные данные и ожидаемый результат (Mathcad) | Полученный результат |
| 1 |  |  |

Продолжение таблицы 7

|  |  |  |
| --- | --- | --- |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

Приложение В

(обязательное)

Исходный код программы с промежуточными результатами

Program DelphiLab1FuncIntermediate;

{

Matrices A and B have order 3. The user fills in matrices

A and B.

The program finds the value of the matrix expression: (A –

B^2) \* (2\*A + B).

}

{$APPTYPE CONSOLE}

Type

TMatrix = Array[1..3, 1..3] of Integer;

// TMatrix - type for a matrix of order 3

Var

MatrixA: TMatrix;

MatrixB: TMatrix;

LeftExpressionPart, LeftExpression: TMatrix;

RightExpressionPart, RightExpression: TMatrix;

ResultOfExpression: TMatrix;

// FirstMatrix, SecondMatrix - matrix A and B respectively

// LeftExpression, RightExpression, ResultOfExpression –

// the result of the matrix expression

// The function finds the sum (difference) of two matrices.

// For the difference, set the third parameter to true

Procedure MatrixSum(Const FirstMatrix, SecondMatrix:

TMatrix; var Result: TMatrix; Const isDifference:

Boolean = False);

Var

i, j, Sign: Integer;

// i,j – cycle counter

// Sign – sign

Begin

// Determine the sign

If isDifference then

Sign:= -1

Else

Sign:=1;

// Sum the matrices

For i:= 1 to 3 do

For j:= 1 to 3 do

Result[i,j]:=FirstMatrix[i,j]+Sign\*SecondMatrix[i,j];

End;

// The function finds the product of a matrix and a number

Procedure MultiplyMatrixAndConst(Const Matrix: TMatrix;

Const Num: Integer; var Result: TMatrix);

Var

i, j:integer;

// i, j - cycle counter

Begin

// Finding the product

For i:= 1 to 3 do

For j:= 1 to 3 do

Result[i,j]:=Num\*Matrix[i,j];

End;

// The function finds the product of two matrices (matrices

// must be the same size)

Procedure MultiplyMatrices(Const FirstMatrix, SecondMatrix:

TMatrix; var Result: TMatrix);

Var

i, j, l: Integer;

// i, j, l - cycle counter

Begin

// Multiplying matrices

For i:= 1 to 3 do

For j:= 1 to 3 do

Begin

// Initializing the value of matrix element

Result[i,j]:= 0;

For l:= 1 to 3 do

Result[i,j]:= Result[i,j] + FirstMatrix[i,l] \*

SecondMatrix[l,j];

End;

End;

// Function to read a matrix

Procedure InputMatrix(var Result: TMatrix);

Var

i,j: integer;

isCorrect: boolean;

// i, j - cycle counter

// isCorrect - flag to confirm the correctness of entering

// numbers

Begin

// Matrix input

for i := 1 to 3 do

begin

for j := 1 to 3 do

repeat

isCorrect:= True;

Try

Read(Result[i, j]);

Except

isCorrect:= False;

Writeln('Matrix value entered incorrectly. Repeat

again');

End;

until isCorrect;

Readln;

end;

End;

// The procedure outputs a matrix

Procedure OutputMatrix(Const Matrix:TMatrix);

Var

i,j:integer;

// i, j - cycle counter

Begin

// Matrix output

For i:= 1 to 3 do

Begin

For j:= 1 to 3 do

Write(Matrix[i,j]:5);

Writeln

End;

Writeln

End;

Begin

// Outputting the initial matrixes

Writeln('Enter matrix A:');

InputMatrix(MatrixA);

Writeln;

Writeln('Enter matrix B:');

InputMatrix(MatrixB);

Writeln;

Writeln('B^2 =');

MultiplyMatrices(MatrixB, MatrixB, LeftExpressionPart);

OutputMatrix(LeftExpressionPart);

Writeln;

Writeln('(A - B^2) =');

MatrixSum(MatrixA, LeftExpressionPart, LeftExpression,

true);

OutputMatrix(LeftExpression);

Writeln;

Writeln('2\*A =');

MultiplyMatrixAndConst(MatrixA, 2, RightExpressionPart);

OutputMatrix(RightExpressionPart);

Writeln;

Writeln('(2\*A + B) =');

MatrixSum(RightExpressionPart, MatrixB, RightExpression);

OutputMatrix(RightExpression);

Writeln;

// Calculating the value of the expression

Writeln('(A - B^2) \* (2\*A + B) =');

MultiplyMatrices(LeftExpression, RightExpression,

ResultOfExpression);

OutputMatrix(ResultOfExpression);

Readln;

End.

Приложение Г

(обязательное)

**Исходный код программы с использованием динамического массива**

Program Normal\_Implementation;

{

Matrices A and B have order 3. The user fills in matrices

A and B.

The program finds the value of the matrix expression: (A –

B^2) \* (2\*A + B).

}

{$APPTYPE CONSOLE}

Type

TMatrix = Array of Array of Double;

// TMatrix - type for a matrix of order 3

Var

MatrixA: TMatrix;

MatrixB: TMatrix;

Res: TMatrix;

MartixOrded: integer;

// FirstMatrix, SecondMatrix - matrix A and B respectively

// Res - the result of the matrix expression

// MartixOrded - matrix order

// The function finds the sum (difference) of two matrices.

// For the difference, set the third parameter to true

Function MatrixSum(Const FirstMatrix, SecondMatrix: TMatrix;

Const isDifference: Boolean = False) : TMatrix;

Var

i, j, Sign: Integer;

// i,j – cycle counter

// Sign – sign

Begin

// Checking for the correctness of matrices

if (Length(FirstMatrix) = Length(SecondMatrix)) and

(Length(FirstMatrix[0]) = Length(SecondMatrix[0]) )

then

begin

// Determine the sign

If isDifference then

Sign:= -1

Else

Sign:= 1;

// Sum the matrices

SetLength(Result, Length(FirstMatrix),

Length(FirstMatrix[0]));

For i:= Low(Result) to High(Result) do

For j:= Low(Result[0]) to High(Result[0]) do

Result[i,j]:=FirstMatrix[i,j] +

Sign\*SecondMatrix[i,j];

end;

End;

// The function finds the product of a matrix and a number

Function MultiplyMatrixAndConst(Const Matrix: TMatrix; Const

Num: Double):TMatrix;

Var

i, j:integer;

// i, j - cycle counter

Begin

// Finding the product

SetLength(Result, Length(Matrix), Length(Matrix[0]));

For i:= Low(Result) to High(Result) do

For j:= Low(Result[0]) to High(Result[0]) do

Result[i,j]:=Num\*Matrix[i,j];

End;

// The function finds the product of two matrices (matrices

// must be the same size)

Function MultiplyMatrices(Const FirstMatrix, SecondMatrix:

TMatrix): TMatrix;

Var

i, j, l: Integer;

// i, j, l - cycle counter

Begin

if Length(FirstMatrix[0]) = Length(SecondMatrix) then

begin

// Multiplying matrices

SetLength(Result, Length(FirstMatrix),

Length(SecondMatrix[0]));

For i:= Low(Result) to High(Result) do

For j:= Low(Result[0]) to High(Result[0]) do

Begin

// Initializing the value of matrix element

Result[i,j]:= 0;

For l:= Low(SecondMatrix) to High(SecondMatrix) do

Result[i,j]:= Result[i,j] + FirstMatrix[i,l] \*

SecondMatrix[l,j];

End;

end;

End;

// Function to read a matrix

Procedure InputMatrix(var Matrix: TMatrix);

Var

i,j: integer;

isCorrect: boolean;

// i, j - cycle counter

// isCorrect - flag to confirm the correctness of entering

// numbers

Begin

// Matrix input

for i := Low(Matrix) to High(Matrix) do

begin

for j := Low(Matrix[0]) to High(Matrix[0]) do

repeat

isCorrect:= True;

Try

Read(Matrix[i, j]);

Except

isCorrect:= False;

Writeln('Matrix value entered incorrectly. Repeat

again');

End;

until isCorrect;

Readln;

end;

End;

// Function to read a matrix order

Function InputMatrixOrder(): Integer;

Var

isCorrect: boolean;

// isCorrect - flag to confirm the correctness of entering numbers

Begin

// Matrix input

repeat

isCorrect:= True;

Try

Read(Result);

Except

isCorrect:= False;

Writeln('Matrix order value entered incorrectly. Repeat

again');

End;

until isCorrect;

End;

// The procedure outputs a matrix

Procedure OutputMatrix(Const Matrix:TMatrix);

Var

i,j:integer;

// i, j - cycle counter

Begin

// Matrix output

For i:= low(Matrix) to High(Matrix) do

Begin

For j:= Low(Matrix[0]) to High(Matrix[0]) do

Write(Matrix[i,j]:8:1);

Writeln

End;

Writeln

End;

Begin

Writeln('Enter the order of matrices');

MartixOrded:= InputMatrixOrder();

Writeln;

// Outputting the initial matrixes

Writeln('Enter matrix A:');

SetLength(MatrixA, MartixOrded, MartixOrded);

InputMatrix(MatrixA);

Writeln;

Writeln('Enter matrix B:');

SetLength(MatrixB, MartixOrded, MartixOrded);

InputMatrix(MatrixB);

Writeln;

// Calculating the value of the expression

Writeln('(A - B^2) \* (2\*A + B)=');

Res:= MultiplyMatrices( MatrixSum(MatrixA,

MultiplyMatrices(MatrixB, MatrixB), true),

MatrixSum(MultiplyMatrixAndConst(MatrixA, 2),

MatrixB) );

OutputMatrix(Res);

Readln;

End.