УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.3

по предмету «Основы алгоритмизации и программирования»

Вариант 15

Выполнила:

Городко К. Е.

Гр. 351005

Проверила:

Данилова Г. В.

Минск 2023

**ЗАДАНИЕ:**

Найти произведение вектора на матрицу.

**КОД ПРОГРАММЫ DELPHI:**

Program LAB2\_3;

Uses

System.SysUtils;

Type

TMatrix = Array Of Array Of Integer;

TVector = Array Of Integer;

Const

MINSIZE = 2;

MAXSIZE = 10;

MIN = -1000;

MAX = 1000;

Function InputSize(OutputMessage: String): Integer;

Var

Size: Integer;

IsCorrect: Boolean;

Begin

Repeat

IsCorrect := True;

Write(OutputMessage);

Try

Readln(Size);

Except

IsCorrect := False;

Writeln('Введенные данные не соответствуют условию. Повторите попытку.');

End;

If IsCorrect And ((Size < MINSIZE) Or (Size > MAXSIZE)) Then

Begin

IsCorrect := False;

Writeln('Введенные данные не соответствуют условию. Повторите попытку.');

End;

Until (IsCorrect);

InputSize := Size;

End;

Function InputMatrix(RowNum, ColNum: Integer): TMatrix;

Var

I, J: Integer;

IsCorrect: Boolean;

Matrix: TMatrix;

Begin

SetLength(Matrix, RowNum, ColNum);

For I := 0 To High(Matrix) Do

For J := 0 To High(Matrix[I]) Do

Repeat

IsCorrect := True;

Write('Введите элемент ', I + 1, ' строки, ', J + 1,

' столбца матрицы: ');

Try

Readln(Matrix[I][J]);

Except

Writeln('Введенные данные не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

If IsCorrect And

((Matrix[I][J] > MAX) Or (Matrix[I][J] < MIN)) Then

Begin

Writeln('Введенные данные не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

Until (IsCorrect);

Writeln;

InputMatrix := Matrix;

End;

Function InputVector(Size: Integer; VectorOrMatrix: String): TVector;

Var

I: Integer;

Vector: TVector;

IsCorrect: Boolean;

Begin

SetLength(Vector, Size);

For I := 0 To High(Vector) Do

Repeat

IsCorrect := True;

Write('Введите ', I + 1, ' элемент ', VectorOrMatrix, ': ');

Try

Readln(Vector[I]);

Except

Writeln('Введенные данные не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

If IsCorrect And ((Vector[I] > MAX) Or (Vector[I] < MIN)) Then

Begin

Writeln('Введенные данные не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

Until (IsCorrect);

Writeln;

InputVector := Vector;

End;

Function CheckFilePath(Path: String): Boolean;

Var

IsCorrect: Boolean;

Begin

If Not FileExists(Path) Then

Begin

Writeln('Введенного файла не существует. Повторите попытку.');

IsCorrect := False;

End

Else If ExtractFileExt(Path) <> '.txt' Then

Begin

Writeln('Введенный Вами файл не является текстовым. Повторите попытку.');

IsCorrect := False;

End;

CheckFilePath := IsCorrect;

End;

Function CheckFileInputPath(MatrixOrVector: String): String;

Var

Path: String;

IsCorrect: Boolean;

Begin

Repeat

IsCorrect := True;

Writeln('Введите путь к файлу, содержащему ', MatrixOrVector, '.');

Readln(Path);

IsCorrect := CheckFilePath(Path);

Until IsCorrect;

CheckFileInputPath := Path;

End;

Function GetSizeFromFile(Path, MatrixOrVector: String): Integer;

Var

Size: Integer;

FIn: TextFile;

IsCorrect: Boolean;

Begin

AssignFile(FIn, Path);

Repeat

IsCorrect := True;

Try

Try

Reset(FIn);

Read(FIn, Size);

If (IsCorrect And (Size < MINSIZE) Or (Size > MAXSIZE)) Then

Begin

Writeln('Размер в выбранном файле не соответствуют условию.

Повторите попытку.');

IsCorrect := False;

End;

Finally

CloseFile(FIn);

End;

Except

Writeln('Данные выбранного файла не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

If Not IsCorrect Then

Path := CheckFileInputPath(MatrixOrVector);

Until IsCorrect;

GetSizeFromFile := Size;

End;

Function FillMatrixFromFile(RowNum, ColNum: Integer; Path: String;

Matrix: TMatrix): TMatrix;

Var

I, J: Integer;

FIn: TextFile;

IsCorrect: Boolean;

Begin

SetLength(Matrix, RowNum, ColNum);

AssignFile(FIn, Path);

Repeat

IsCorrect := True;

Try

Try

Reset(FIn);

Readln(FIn);

For I := 0 To High(Matrix) Do

For J := 0 To High(Matrix[I]) Do

Begin

Read(FIn, Matrix[I][J]);

If (IsCorrect And ((Matrix[I][J] < MIN) Or

(Matrix[I][J] > MAX))) Then

Begin

Writeln('Данные выбранного файла не соответствуют

условию. Повторите попытку.');

IsCorrect := False;

End;

End;

If (IsCorrect And (Not Eof(FIn))) Then

Begin

Writeln('Размер введенной матрицы не соответствует заданному.

Повторите попытку.');

IsCorrect := False;

End;

Finally

CloseFile(FIn);

End;

Except

Writeln('Данные выбранного файла не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

If Not IsCorrect Then

Path := CheckFileInputPath('матрицу');

Until IsCorrect;

Writeln('Данные из файла успешно считаны.');

Writeln;

FillMatrixFromFile := Matrix;

End;

Function FillVectorFromFile(Size: Integer; Path, MatrixOrVector: String): TVector;

Var

I: Integer;

Vector: TVector;

FIn: TextFile;

IsCorrect: Boolean;

Begin

SetLength(Vector, Size);

AssignFile(FIn, Path);

Repeat

IsCorrect := True;

Try

Try

Reset(FIn);

Readln(FIn);

For I := 0 To High(Vector) Do

Begin

Read(FIn, Vector[I]);

If (IsCorrect And ((Vector[I] < MIN) Or

(Vector[I] > MAX))) Then

Begin

Writeln('Данные выбранного файла не соответствуют условию.

Повторите попытку.');

IsCorrect := False;

End;

End;

If (IsCorrect And (Not Eof(FIn))) Then

Begin

Writeln('Размер введенного вектора/матрицы не соответствует

заданному. Повторите попытку.');

IsCorrect := False;

End;

Finally

CloseFile(FIn);

End;

Except

Writeln('Данные выбранного файла не соответствуют условию. Повторите

попытку.');

IsCorrect := False;

End;

If Not IsCorrect Then

Path := CheckFileInputPath(MatrixOrVector);

Until IsCorrect;

Writeln('Данные из файла успешно считаны.');

Writeln;

FillVectorFromFile := Vector;

End;

Procedure OutputMatrix(Matrix: TMatrix);

Var

I, J: Integer;

Begin

For I := 0 To High(Matrix) Do

Begin

For J := 0 To High(Matrix[I]) Do

Write(Matrix[I][J], ' ');

Writeln;

End;

End;

Procedure OutputVectorColumn(Vector: TVector);

Var

I: Integer;

Begin

For I := 0 To High(Vector) Do

Writeln(Vector[I]);

Writeln;

End;

Procedure OutputVectorRow(Vector: TVector);

Var

I: Integer;

Begin

For I := 0 To High(Vector) Do

Write(Vector[I], ' ');

Writeln;

Writeln;

End;

Function FindProductVector(Vector: TVector; Matrix: TMatrix): TVector;

Var

I, J: Integer;

ProductVector: TVector;

Begin

SetLength(ProductVector, (High(Matrix[0]) + 1));

For I := 0 To High(ProductVector) Do

Begin

ProductVector[I] := 0;

For J := 0 To High(Matrix) Do

ProductVector[I] := ProductVector[I] + (Matrix[J][I] \* Vector[J]);

End;

FindProductVector := ProductVector;

End;

Function FindProductMatrix(RowNum, ColNum: Integer;

Vector, Matrix: TVector): TMatrix;

Var

I, J: Integer;

ProductMatrix: TMatrix;

Begin

SetLength(ProductMatrix, RowNum, ColNum);

For I := 0 To High(ProductMatrix) Do

For J := 0 To High(ProductMatrix[I]) Do

ProductMatrix[I][J] := Vector[I] \* Matrix[J];

FindProductMatrix := ProductMatrix;

End;

Function CheckFileOutputPath(): String;

Var

Path: String;

IsCorrect: Boolean;

Begin

Writeln;

Repeat

IsCorrect := True;

Writeln('Введите путь к файлу, в который нужно записать результат.');

Readln(Path);

IsCorrect := CheckFilePath(Path);

If IsCorrect And FileIsReadOnly(Path) Then

Begin

Writeln('Введенный Вами файл доступен только для чтения. Повторите

попытку.');

IsCorrect := False;

End;

Until IsCorrect;

CheckFileOutputPath := Path;

End;

Procedure WriteVectorIntoFile(Path: String; ProductVector: TVector);

Var

I: Integer;

IsCorrect: Boolean;

FOut: TextFile;

Begin

Repeat

IsCorrect := True;

Assign(FOut, Path);

Try

Try

Rewrite(FOut);

For I := 0 To High(ProductVector) Do

Writeln(FOut, ProductVector[I]);

Finally

CloseFile(FOut);

End;

Except

Writeln('Произошла ошибка. Повторите попытку.');

IsCorrect := False;

Path := CheckFileOutputPath();

End;

Until IsCorrect;

Writeln('Результат записан.');

End;

Procedure WriteMatrixIntoFile(Path: String; ProductMatrix: TMatrix);

Var

I, J: Integer;

IsCorrect: Boolean;

FOut: TextFile;

Begin

Repeat

IsCorrect := True;

Assign(FOut, Path);

Try

Try

Rewrite(FOut);

For I := 0 To High(ProductMatrix) Do

Begin

For J := 0 To High(ProductMatrix[I]) Do

Write(FOut, ProductMatrix[I][J], ' ');

Writeln(FOut);

End;

Finally

CloseFile(FOut);

End;

Except

Writeln('Произошла ошибка. Повторите попытку.');

IsCorrect := False;

Path := CheckFileOutputPath();

End;

Until IsCorrect;

Writeln('Результат записан.');

End;

Function GetVectorFromFile(MatrixOrVector: String): TVector;

Var

Size: Integer;

FInPathVector: String;

Vector: TVector;

Begin

FInPathVector := CheckFileInputPath(MatrixOrVector);

Size := GetSizeFromFile(FInPathVector, MatrixOrVector);

Vector := FillVectorFromFile(Size, FInPathVector, MatrixOrVector);

GetVectorFromFile := Vector;

End;

Function GetMatrixFromFile(RowNum: Integer): TMatrix;

Var

ColNum: Integer;

FInPathMatrix: String;

Matrix: TMatrix;

Begin

Writeln('Количество строк матрицы равно ', RowNum, '. В первой строке файла

введите количество столбцов.');

FInPathMatrix := CheckFileInputPath('матрицу');

ColNum := GetSizeFromFile(FInPathMatrix, 'матрицу');

Matrix := FillMatrixFromFile(RowNum, ColNum, FInPathMatrix, Matrix);

GetMatrixFromFile := Matrix;

End;

Procedure OutputVectors(VectorColumn, VectorRow: TVector);

Begin

Writeln('Введенный вектор:');

OutputVectorColumn(VectorColumn);

Writeln('Введенная матрица:');

OutputVectorRow(VectorRow);

End;

Procedure OutputVectorAndMatrix(Vector: TVector; Matrix: TMatrix);

Begin

Writeln('Введенный вектор:');

OutputVectorRow(Vector);

Writeln('Введенная матрица:');

OutputMatrix(Matrix);

End;

Procedure InputFromConsole(VectorType: Integer);

Var

RowNum, ColNum: Integer;

Vector, VectorRow, ProductVector: TVector;

Matrix, ProductMatrix: TMatrix;

Begin

RowNum := InputSize('Введите размер вектора (от 2 до 10): ');

Vector := InputVector(RowNum, 'вектора');

ColNum := InputSize('Введите количество столбцов матрицы (от 2 до 10): ');

If VectorType = 0 Then

Begin

VectorRow := InputVector(ColNum, 'матрицы');

OutputVectors(Vector, VectorRow);

ProductMatrix := FindProductMatrix(High(Vector) + 1,

High(VectorRow) + 1, Vector, VectorRow);

Writeln('Результат произведения вектора и матрицы:');

OutputMatrix(ProductMatrix);

End

Else

Begin

Writeln('Количество строк матрицы равно ', RowNum, '.');

Matrix := InputMatrix(RowNum, ColNum);

OutputVectorAndMatrix(VectorRow, Matrix);

ProductVector := FindProductVector(VectorRow, Matrix);

Writeln('Результат произведения вектора и матрицы:');

OutputVectorRow(ProductVector);

End;

End;

Procedure InputFromFile(VectorType: Integer);

Var

RowNum, ColNum: Integer;

Vector, VectorRow, ProductVector: TVector;

Matrix, ProductMatrix: TMatrix;

FOutPath: String;

Begin

Writeln('В первой строке файла должны быть записаны размеры (от 2 до 10).');

Vector := GetVectorFromFile('вектор');

If VectorType = 0 Then

Begin

VectorRow := GetVectorFromFile('матрицу (вектор-строку)');

OutputVectors(Vector, VectorRow);

ProductMatrix := FindProductMatrix(High(Vector) + 1, High(VectorRow) + 1,

Vector, VectorRow);

FOutPath := CheckFileOutputPath();

WriteMatrixIntoFile(FOutPath, ProductMatrix);

End

Else

Begin

Matrix := GetMatrixFromFile(High(Vector) + 1);

OutputVectorAndMatrix(Vector, Matrix);

ProductVector := FindProductVector(Vector, Matrix);

FOutPath := CheckFileOutputPath();

WriteVectorIntoFile(FOutPath, ProductVector);

End;

End;

Function CheckChoiceInput(): Integer;

Var

Num: Integer;

IsCorrect: Boolean;

Begin

Repeat

IsCorrect := True;

Try

Readln(Num);

Except

IsCorrect := False;

Writeln('Введенные данные не соответствуют условию. Повторите попытку.');

End;

If IsCorrect And ((Num <> 0) And (Num <> 1)) Then

Begin

IsCorrect := False;

Writeln('Введенные данные не соответствуют условию. Повторите попытку.');

End;

Until (IsCorrect);

CheckChoiceInput := Num;

End;

Function ChooseVectorType(): Integer;

Var

VectorType: Integer;

Begin

Writeln('Введите 0, если требуется умножить вектор-столбец на матрицу, и 1, если

вектор-строку.');

VectorType := CheckChoiceInput();

If VectorType = 0 Then

Writeln('Для умножения вектора-столбца на матрицу, матрица должна состоять из

1 строки.')

Else

Writeln('Для умножения вектора-строки на матрицу, число столбцов вектора

должно совпадать с числом строк матрицы.');

Writeln;

ChooseVectorType := VectorType;

End;

Procedure WriteCondition();

Begin

Writeln('Данная программа находит произведение вектора на матрицу.');

Writeln('Элементы вектора и матрицы - целые числа от -1000 до 1000.');

End;

Var

Choice, VectorType: Integer;

Begin

WriteCondition();

Writeln('Если Вы хотите вводить данные в консоль, введите 0. Если использовать

файл, введите 1.');

Choice := CheckChoiceInput();

VectorType := ChooseVectorType();

If Choice = 0 Then

InputFromConsole(VectorType)

Else

InputFromFile(VectorType);

End.

**КОД ПРОГРАММЫ С++:**

#include <iostream>

#include <fstream>

using namespace std;

const int MINSIZE = 2;

const int MAXSIZE = 10;

const int MIN = -1000;

const int MAX = 1000;

int inputSize(string outputMessage) {

int size = 0;

bool isIncorrect;

do {

isIncorrect = false;

cout << outputMessage;

cin >> size;

if (cin.get() != '\n') {

cin.clear();

while (cin.get() != '\n');

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите попытку."

<< endl;

}

if (!isIncorrect && ((size < MINSIZE) || (size > MAXSIZE))) {

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите попытку."

<< endl;

}

} while (isIncorrect);

return size;

}

int\*\* createMatrix(int rowNum, int colNum) {

int\*\* matrix = new int\* [rowNum];

for (int i = 0; i < rowNum; i++)

matrix[i] = new int[colNum];

return matrix;

}

int\*\* inputMatrix(int rowNum, int colNum) {

bool isIncorrect;

int\*\* matrix = createMatrix(rowNum, colNum);

for (int i = 0; i < rowNum; i++)

for (int j = 0; j < colNum; j++)

do {

isIncorrect = false;

cout << "Введите элемент " << i + 1 << " строки, " << j + 1 << "

столбца матрицы : ";

cin >> matrix[i][j];

if (!isIncorrect && cin.get() != '\n') {

cin.clear();

while (cin.get() != '\n');

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите

попытку." << endl;

}

if (!isIncorrect && (matrix[i][j] < MIN || matrix[i][j] > MAX)) {

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите

попытку." << endl;

}

} while (isIncorrect);

cout << endl;

return matrix;

}

int\* inputVector(int size, string vectorOrMatrix) {

bool isIncorrect;

int\* vector = new int[size];

for (int i = 0; i < size; i++)

do {

isIncorrect = false;

cout << "Введите " << i + 1 << " элемент " << vectorOrMatrix << ": ";

cin >> vector[i];

if (!isIncorrect && cin.get() != '\n') {

cin.clear();

while (cin.get() != '\n');

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите

попытку." << endl;

}

if (!isIncorrect && (vector[i] < MIN || vector[i] > MAX)) {

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите

попытку." << endl;

}

} while (isIncorrect);

cout << endl;

return vector;

}

bool checkFileExtension(string path) {

bool isIncorrect = true;

if (path.substr(path.size() - 4) == ".txt")

isIncorrect = false;

else

cout << "Введенный Вами файл не является текстовым. Повторите попытку."

<< endl;

return isIncorrect;

}

string checkFileInputPath(string matrixOrVector) {

string path;

bool isIncorrect;

do {

isIncorrect = false;

cout << "Введите путь к файлу, содержащему " << matrixOrVector << "."

<< endl;

cin >> path;

ifstream fin(path);

fin.open(path);

if (!fin.is\_open()) {

cout << "Введенного файла не существует. Повторите попытку." << endl;

isIncorrect = true;

}

if (!isIncorrect)

isIncorrect = checkFileExtension(path);

fin.close();

} while (isIncorrect);

return path;

}

int getSizeFromFile(string path, string matrixOrVector) {

int size = 0;

bool isIncorrect;

do {

isIncorrect = false;

ifstream fin(path);

try {

fin >> size;

}

catch (string errorMessage)

{

cout << "Данные выбранного файла не соответствуют условию. Повторите

попытку." << endl;

isIncorrect = true;

}

if (!isIncorrect && (size < MINSIZE || size > MAXSIZE))

{

cout << "Размер в выбранном файле не соответствуют условию. Повторите

попытку." << endl;

isIncorrect = true;

}

fin.close();

if (isIncorrect)

path = checkFileInputPath(matrixOrVector);

} while (isIncorrect);

return size;

}

int\*\* fillMatrixFromFile(string path, int rowNum, int colNum) {

bool isIncorrect;

int\*\* matrix = createMatrix(rowNum, colNum);

do {

isIncorrect = false;

ifstream fin(path);

fin >> colNum;

try {

for (int i = 0; i < rowNum; i++)

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

fin >> matrix[i][j];

if (!isIncorrect && ((matrix[i][j] < MIN) ||

(matrix[i][j] > MAX))) {

cout << "Данные выбранного файла не соответствуют условию.

Повторите попытку." << endl;

isIncorrect = true;

}

}

if (!isIncorrect && !fin.eof()) {

cout << "Размер введенной матрицы не соответствует заданному.

Повторите попытку." << endl;

isIncorrect = true;

}

}

catch (string errorMessage)

{

cout << "Данные выбранного файла не соответствуют условию. Повторите

попытку." << endl;

isIncorrect = true;

}

fin.close();

if (isIncorrect)

path = checkFileInputPath("матрицу");

} while (isIncorrect);

cout << "Данные из файла успешно считаны." << endl;

cout << endl;

return matrix;

}

int\* fillVectorFromFile(int size, string path, string matrixOrVector) {

bool isIncorrect;

int\* vector = new int[size];

do {

isIncorrect = false;

ifstream fin(path);

fin >> size;

try {

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

fin >> vector[i];

if (!isIncorrect && ((vector[i] < MIN) || (vector[i] > MAX))) {

cout << "Данные выбранного файла не соответствуют условию.

Повторите попытку." << endl;

isIncorrect = true;

}

}

if (!isIncorrect && !fin.eof()) {

cout << "Размер введенного вектора/матрицы не соответствует

заданному. Повторите попытку." << endl;

isIncorrect = true;

}

}

catch (string errorMessage)

{

cout << "Данные выбранного файла не соответствуют условию. Повторите

попытку." << endl;

isIncorrect = true;

}

fin.close();

if (isIncorrect)

path = checkFileInputPath(matrixOrVector);

} while (isIncorrect);

cout << "Данные из файла успешно считаны." << endl;

cout << endl;

return vector;

}

void outputMatrix(int rowNum, int colNum, int\*\* matrix) {

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

for (int j = 0; j < colNum; j++)

cout << matrix[i][j] << " ";

cout << endl;

}

}

void outputVectorColumn(int size, int\* vector) {

for (int i = 0; i < size; i++)

cout << vector[i] << endl;

cout << endl;

}

void outputVectorRow(int size, int\* vector) {

for (int i = 0; i < size; i++)

cout << vector[i] << " ";

cout << endl;

cout << endl;

}

int\* findProductVector(int rowNum, int colNum, int\* vector, int\*\* matrix) {

int\* productVector = new int[colNum];

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

productVector[i] = 0;

for (int j = 0; j < rowNum; j++)

productVector[i] = productVector[i] + (matrix[j][i] \* vector[j]);

}

return productVector;

}

int\*\* findProductMatrix(int rowNum, int colNum, int\* vector, int\* matrix) {

int\*\* productMatrix = createMatrix(rowNum, colNum);

for (int i = 0; i < rowNum; i++)

for (int j = 0; j < colNum; j++)

productMatrix[i][j] = matrix[j] \* vector[i];

return productMatrix;

}

string checkFileOutputPath() {

string path;

bool isIncorrect;

cout << endl;

do {

isIncorrect = false;

cout << "Введите путь к файлу, в который нужно записать результат." << endl;

cin >> path;

ofstream fout(path);

fout.open(path);

if (!fout.is\_open()) {

cout << "Произошла ошибка. Повторите попытку." << endl;

isIncorrect = true;

}

if (!isIncorrect)

isIncorrect = checkFileExtension(path);

fout.close();

} while (isIncorrect);

cout << endl;

return path;

}

void writeVectorIntoFile(string path, int size, int\* productVector) {

bool isIncorrect;

do {

isIncorrect = false;

ofstream fout(path);

try {

for (int i = 0; i < size; i++)

fout << productVector[i] << endl;

}

catch (string errorMessage)

{

cout << "Произошла ошибка. Повторите попытку." << endl;

isIncorrect = true;

}

fout.close();

if (isIncorrect)

path = checkFileOutputPath();

} while (isIncorrect);

cout << "Результат записан.";

}

void writeMatrixIntoFile(string path, int rowNum, int colNum, int\*\* productMatrix) {

bool isIncorrect;

do {

isIncorrect = false;

ofstream fout(path);

try {

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

for (int j = 0; j < colNum; j++)

fout << productMatrix[i][j] << " ";

fout << endl;

}

}

catch (string errorMessage)

{

cout << "Произошла ошибка. Повторите попытку." << endl;

isIncorrect = true;

}

fout.close();

if (isIncorrect)

path = checkFileOutputPath();

} while (isIncorrect);

cout << "Результат записан.";

}

void outputVectors(int rowNum, int colNum, int\* vectorColumn, int\* vectorRow) {

cout << "Введенный вектор:" << endl;

outputVectorColumn(rowNum, vectorColumn);

cout << "Введенная матрица:" << endl;

outputVectorRow(colNum, vectorRow);

}

void outputVectorAndMatrix(int rowNum, int colNum, int\* vectorRow, int\*\* matrix) {

cout << "Введенный вектор:" << endl;

outputVectorRow(rowNum, vectorRow);

cout << "Введенная матрица:" << endl;

outputMatrix(rowNum, colNum, matrix);

}

void inputFromConsole(int vectorType) {

int rowNum = 0;

int colNum = 0;

rowNum = inputSize("Введите размер вектора (от 2 до 10): ");

int\* vector = inputVector(rowNum, "вектора");

colNum = inputSize("Введите количество столбцов матрицы (от 2 до 10): ");

if (vectorType == 0) {

int\* vectorRow = inputVector(colNum, "матрицы");

outputVectors(rowNum, colNum, vector, vectorRow);

int\*\* productMatrix = findProductMatrix(rowNum, colNum, vector, vectorRow);

cout << "Результат произведения вектора и матрицы:" << endl;

outputMatrix(rowNum, colNum, productMatrix);

delete[] vectorRow;

delete[] productMatrix;

}

else {

cout << "Количество строк матрицы равно " << rowNum << "." << endl;

int\*\* matrix = inputMatrix(rowNum, colNum);

outputVectorAndMatrix(rowNum, colNum, vector, matrix);

int\* productVector = findProductVector(rowNum, colNum, vector, matrix);

cout << "Результат произведения вектора и матрицы:" << endl;

outputVectorRow(colNum, productVector);

delete[] matrix;

delete[] productVector;

}

delete[] vector;

}

void inputFromFile(int vectorType) {

int rowNum = 0;

int colNum = 0;

string finPathVector;

string finPathMatrix;

string foutPath;

cout << "В первой строке файла должны быть записаны размеры (от 2 до 10)."

<< endl;

finPathVector = checkFileInputPath("вектор");

rowNum = getSizeFromFile(finPathVector, "вектор");

int\* vector = fillVectorFromFile(rowNum, finPathVector, "вектор");

if (vectorType == 1)

cout << "Количество строк матрицы равно " << rowNum << ". В первой строке

файла введите количество столбцов." << endl;

finPathMatrix = checkFileInputPath("матрицу");

colNum = getSizeFromFile(finPathMatrix, "матрицу");

if (vectorType == 0) {

int\* vectorRow = fillVectorFromFile(colNum, finPathMatrix, "матрицу (вектор-

строку)");

outputVectors(rowNum, colNum, vector, vectorRow);

int\*\* productMatrix = findProductMatrix(rowNum, colNum, vector, vectorRow);

foutPath = checkFileOutputPath();

writeMatrixIntoFile(foutPath, rowNum, colNum, productMatrix);

delete[] vectorRow;

delete[] productMatrix;

}

else {

int\*\* matrix = fillMatrixFromFile(finPathMatrix, rowNum, colNum);

outputVectorAndMatrix(rowNum, colNum, vector, matrix);

int\* productVector = findProductVector(rowNum, colNum, vector, matrix);

foutPath = checkFileOutputPath();

writeVectorIntoFile(foutPath, colNum, productVector);

delete[] matrix;

delete[] productVector;

}

delete[] vector;

}

int checkChoiceInput() {

int num = 0;

bool isIncorrect;

do {

isIncorrect = false;

cin >> num;

if (cin.get() != '\n') {

cin.clear();

while (cin.get() != '\n');

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите попытку."

<< endl;

}

if (!isIncorrect && (num != 0) && (num != 1)) {

isIncorrect = true;

cout << "Введенные данные не соответствуют условию. Повторите попытку."

<< endl;

}

} while (isIncorrect);

return num;

}

int chooseVectorType() {

int vectorType = 0;

cout << "Введите 0, если требуется умножить вектор-столбец на матрицу, и 1, если

вектор-строку." << endl;

vectorType = checkChoiceInput();

if (vectorType == 0)

cout << "Для умножения вектора-столбца на матрицу, матрица должна состоять из

1 строки." << endl;

else

cout << "Для умножения вектора-строки на матрицу, число столбцов вектора

должно совпадать с числом строк матрицы." << endl;

cout << endl;

return vectorType;

}

void writeCondition() {

cout << "Данная программа находит произведение вектора на матрицу." << endl;

cout << "Элементы вектора и матрицы - целые числа от -1000 до 1000." << endl;

}

int main() {

setlocale(LC\_ALL, "Russian");

int choice = 0;

int vectorType = 0;

writeCondition();

cout << "Если Вы хотите вводить данные в консоль, введите 0. Если использовать

файл, введите 1." << endl;

choice = checkChoiceInput();

vectorType = chooseVectorType();

if (choice == 0)

inputFromConsole(vectorType);

else

inputFromFile(vectorType);

return 0;

}

**КОД ПРОГРАММЫ JAVA:**

import java.util.Scanner;

import java.io.\*;

public class main {

static final int MINSIZE = 2;

static final int MAXSIZE = 10;

static final int MIN = -1000;

static final int MAX = 1000;

private static int inputSize(String outputMessage, Scanner scan) {

int size = 0;

boolean isIncorrect;

do {

isIncorrect = false;

System.out.print(outputMessage);

try {

size = Integer.parseInt(scan.nextLine());

}

catch (NumberFormatException e){

System.out.println("Введенные данные не соответствуют условию.

Повторите попытку.");

isIncorrect = true;

}

if (!isIncorrect && ((size < MINSIZE) || (size > MAXSIZE))) {

isIncorrect = true;

System.out.println("Введенные данные не соответствуют условию.

Повторите попытку.");

}

} while (isIncorrect);

return size;

}

private static int[][] inputMatrix(int rowNum, int colNum, Scanner scan) {

boolean isIncorrect;

int[][] matrix = new int[rowNum][colNum];

for (int i = 0; i < rowNum; i++)

for (int j = 0; j < colNum; j++)

do {

isIncorrect = false;

System.out.print("Введите элемент " + (i + 1) + " строки, " +

(j + 1) + " столбца матрицы : ");

try {

matrix[i][j] = Integer.parseInt(scan.nextLine());

}

catch (NumberFormatException e){

System.out.println("Введенные данные не соответствуют

условию. Повторите попытку.");

isIncorrect = true;

}

if (!isIncorrect && (matrix[i][j] < MIN || matrix[i][j] > MAX)) {

isIncorrect = true;

System.out.println("Введенные данные не соответствуют

условию. Повторите попытку.");

}

} while (isIncorrect);

System.out.println();

return matrix;

}

private static int[] inputVector(int size, String vectorOrMatrix, Scanner scan) {

boolean isIncorrect;

int[] vector = new int[size];

for (int i = 0; i < size; i++)

do {

isIncorrect = false;

System.out.print("Введите " + (i + 1) + " элемент " +

vectorOrMatrix + ": ");

try {

vector[i] = Integer.parseInt(scan.nextLine());

}

catch (NumberFormatException e){

System.out.println("Введенные данные не соответствуют условию.

Повторите попытку.");

isIncorrect = true;

}

if (!isIncorrect && (vector[i] < MIN || vector[i] > MAX)) {

isIncorrect = true;

System.out.println("Введенные данные не соответствуют условию.

Повторите попытку.");

}

} while (isIncorrect);

System.out.println();

return vector;

}

private static boolean checkFilePath(String path, File file) {

boolean isIncorrect = false;

if (!file.exists()) {

System.out.println("Введенного файла не существует. Повторите попытку.");

isIncorrect = true;

}

else if (!path.endsWith(".txt")) {

System.out.println("Введенный Вами файл не является текстовым. Повторите

попытку.");

isIncorrect = true;

}

return isIncorrect;

}

private static String checkFileInputPath(String matrixOrVector, Scanner scan) {

String path;

boolean isIncorrect;

do {

isIncorrect = false;

System.out.println("Введите путь к файлу, содержащему " +

matrixOrVector + ".");

path = String.format(scan.nextLine());

File file = new File(path);

isIncorrect = checkFilePath(path, file);

} while (isIncorrect);

return path;

}

private static int getSizeFromFile(String path, String matrixOrVector,

Scanner scan) {

int size = 0;

boolean isIncorrect;

do {

isIncorrect = false;

try (BufferedReader br = new BufferedReader(new FileReader(path))) {

Scanner scanFile = new Scanner(br);

size = scanFile.nextInt();

scanFile.close();

} catch (Exception e) {

System.out.println("Данные выбранного файла не соответствуют условию.

Повторите попытку.");

isIncorrect = true;

}

if (!isIncorrect && (size < MINSIZE || size > MAXSIZE)) {

System.out.println("Размер в выбранном файле не соответствуют

условию. Повторите попытку.");

isIncorrect = true;

}

if (isIncorrect)

path = checkFileInputPath(matrixOrVector, scan);

} while (isIncorrect);

return size;

}

private static int[][] fillMatrixFromFile(String path, int rowNum, int colNum,

Scanner scan) {

boolean isIncorrect;

int[][] matrix = new int[rowNum][colNum];

do {

isIncorrect = false;

try (BufferedReader br = new BufferedReader(new FileReader(path))) {

Scanner scanFile = new Scanner(br);

scanFile.nextInt();

for (int i = 0; i < rowNum; i++)

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

matrix[i][j] = scanFile.nextInt();

if (!isIncorrect && ((matrix[i][j] < MIN) ||

(matrix[i][j] > MAX))) {

System.out.println("Данные выбранного файла не

соответствуют условию. Повторите

попытку.");

isIncorrect = true;

}

}

if (!isIncorrect & scanFile.hasNext()) {

System.out.println("Размер введенной матрицы не соответствует

заданному. Повторите попытку.");

isIncorrect = true;

}

scanFile.close();

} catch (Exception e) {

System.out.println("Данные выбранного файла не соответствуют условию.

Повторите попытку.");

isIncorrect = true;

}

if (isIncorrect)

path = checkFileInputPath("матрицу", scan);

} while (isIncorrect);

System.out.println("Данные из файла успешно считаны.");

System.out.println();

return matrix;

}

private static int[] fillVectorFromFile(int size, String path, String

matrixOrVector, Scanner scan) {

boolean isIncorrect;

int[] vector = new int[size];

do {

isIncorrect = false;

try (BufferedReader br = new BufferedReader(new FileReader(path))) {

Scanner scanFile = new Scanner(br);

scanFile.nextInt();

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

vector[i] = scanFile.nextInt();

if (!isIncorrect && ((vector[i] < MIN) || (vector[i] > MAX))) {

System.out.println("Данные выбранного файла не соответствуют

условию. Повторите попытку.");

isIncorrect = true;

}

}

if (!isIncorrect & scanFile.hasNext()) {

System.out.println("Размер введенного вектора/матрицы не

соответствует заданному. Повторите попытку.");

isIncorrect = true;

}

scanFile.close();

} catch (Exception e) {

System.out.println("Данные выбранного файла не соответствуют условию.

Повторите попытку.");

isIncorrect = true;

}

if (isIncorrect)

path = checkFileInputPath(matrixOrVector, scan);

} while (isIncorrect);

System.out.println("Данные из файла успешно считаны.");

System.out.println();

return vector;

}

private static void outputMatrix(int rowNum, int colNum, int[][] matrix) {

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

for (int j = 0; j < colNum; j++)

System.out.print(matrix[i][j] + " ");

System.out.println();

}

}

private static void outputVectorColumn(int size, int[] vector) {

for (int i = 0; i < size; i++)

System.out.println(vector[i]);

System.out.println();

}

private static void outputVectorRow(int size, int[] vector) {

for (int i = 0; i < size; i++)

System.out.print(vector[i] + " ");

System.out.println();

System.out.println();

}

private static int[] findProductVector(int rowNum, int colNum, int[] vector,

int[][] matrix) {

int[] productVector = new int[colNum];

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

productVector[i] = 0;

for (int j = 0; j < rowNum; j++)

productVector[i] = productVector[i] + (matrix[j][i] \* vector[j]);

}

return productVector;

}

private static int[][] findProductMatrix(int rowNum, int colNum, int[] vector,

int[] matrix) {

int[][] productMatrix = new int[rowNum][colNum];

for (int i = 0; i < rowNum; i++)

for (int j = 0; j < colNum; j++)

productMatrix[i][j] = matrix[j] \* vector[i];

return productMatrix;

}

private static String checkFileOutputPath(Scanner scan) {

String path;

boolean isIncorrect;

System.out.println();

do {

isIncorrect = false;

System.out.println("Введите путь к файлу, в который нужно записать

результат.");

path = String.format(scan.nextLine());

File file = new File(path);

isIncorrect = checkFilePath(path, file);

if (!isIncorrect & !file.canWrite()) {

System.out.println("Введенный Вами файл доступен только для чтения.

Повторите попытку.");

isIncorrect = true;

}

} while (isIncorrect);

System.out.println();

return path;

}

private static void writeVectorIntoFile(String path, int size,

int[] productVector, Scanner scan) {

boolean isIncorrect;

do {

isIncorrect = false;

try (FileWriter fw = new FileWriter(new File(path))) {

for (int i = 0; i < size; i++)

fw.write(productVector[i] + "\n");

} catch (Exception e) {

System.out.println("Произошла ошибка. Повторите попытку.");

isIncorrect = true;

}

if (isIncorrect)

path = checkFileOutputPath(scan);

} while (isIncorrect);

System.out.println("Результат записан.");

}

private static void writeMatrixIntoFile(String path, int rowNum, int colNum,

int[][] productMatrix, Scanner scan) {

boolean isIncorrect;

do {

isIncorrect = false;

try (FileWriter fw = new FileWriter(new File(path))) {

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

for (int j = 0; j < colNum; j++)

fw.write(productMatrix[i][j] + " ");

fw.write("\n");

}

} catch (Exception e) {

System.out.println("Произошла ошибка. Повторите попытку.");

isIncorrect = true;

}

if (isIncorrect)

path = checkFileOutputPath(scan);

} while (isIncorrect);

System.out.println("Результат записан.");

}

private static void outputVectors(int rowNum, int colNum, int[] vectorColumn,

int[] vectorRow) {

System.out.println("Введенный вектор:");

outputVectorColumn(rowNum, vectorColumn);

System.out.println("Введенная матрица:");

outputVectorRow(colNum, vectorRow);

}

private static void outputVectorAndMatrix(int rowNum, int colNum,

int[] vectorRow, int[][] matrix) {

System.out.println("Введенный вектор:");

outputVectorRow(rowNum, vectorRow);

System.out.println("Введенная матрица:");

outputMatrix(rowNum, colNum, matrix);

}

private static void inputFromConsole(int vectorType, Scanner scan) {

int rowNum = 0;

int colNum = 0;

rowNum = inputSize("Введите размер вектора (от 2 до 10): ", scan);

int[] vector = inputVector(rowNum, "вектора", scan);

colNum = inputSize("Введите количество столбцов матрицы (от 2 до 10): ",

scan);

if (vectorType == 0) {

int[] vectorRow = inputVector(colNum, "матрицы", scan);

outputVectors(rowNum, colNum, vector, vectorRow);

int[][] productMatrix = findProductMatrix(rowNum, colNum, vector,

vectorRow);

System.out.println("Результат произведения вектора и матрицы:");

outputMatrix(rowNum, colNum, productMatrix);

}

else {

System.out.println("Количество строк матрицы равно " + rowNum + ".");

int[][] matrix = inputMatrix(rowNum, colNum, scan);

outputVectorAndMatrix(rowNum, colNum, vector, matrix);

int[] productVector = findProductVector(rowNum, colNum, vector, matrix);

System.out.println("Результат произведения вектора и матрицы:");

outputVectorRow(colNum, productVector);

}

}

private static void inputFromFile(int vectorType, Scanner scan) {

int rowNum = 0;

int colNum = 0;

String finPathVector;

String finPathMatrix;

String foutPath;

System.out.println("В первой строке файла должны быть записаны размеры (от 2

до 10).");

finPathVector = checkFileInputPath("вектор", scan);

rowNum = getSizeFromFile(finPathVector, "вектор", scan);

int[] vector = fillVectorFromFile(rowNum, finPathVector, "вектор", scan);

if (vectorType == 1)

System.out.println("Количество строк матрицы равно " + rowNum + ". В

первой строке файла введите количество столбцов.");

finPathMatrix = checkFileInputPath("матрицу", scan);

colNum = getSizeFromFile(finPathMatrix, "матрицу", scan);

if (vectorType == 0) {

int[] vectorRow = fillVectorFromFile(colNum, finPathMatrix, "матрицу

(вектор-строку)", scan);

outputVectors(rowNum, colNum, vector, vectorRow);

int[][] productMatrix = findProductMatrix(rowNum, colNum, vector,

vectorRow);

foutPath = checkFileOutputPath(scan);

writeMatrixIntoFile(foutPath, rowNum, colNum, productMatrix, scan);

}

else {

int[][] matrix = fillMatrixFromFile(finPathMatrix, rowNum, colNum, scan);

outputVectorAndMatrix(rowNum, colNum, vector, matrix);

int[] productVector = findProductVector(rowNum, colNum, vector, matrix);

foutPath = checkFileOutputPath(scan);

writeVectorIntoFile(foutPath, colNum, productVector, scan);

}

}

private static int checkChoiceInput(Scanner scan) {

int num = 0;

boolean isIncorrect;

do {

isIncorrect = false;

try {

num = Integer.parseInt(scan.nextLine());

}

catch (NumberFormatException e){

System.out.println("Введенные данные не соответствуют условию.

Повторите попытку.");

isIncorrect = true;

}

if (!isIncorrect && (num != 0) && (num != 1)) {

isIncorrect = true;

System.out.println("Введенные данные не соответствуют условию.

Повторите попытку.");

}

} while (isIncorrect);

return num;

}

private static int chooseVectorType(Scanner scan) {

int vectorType = 0;

System.out.println("Введите 0, если требуется умножить вектор-столбец на

матрицу, и 1, если вектор-строку.");

vectorType = checkChoiceInput(scan);

if (vectorType == 0)

System.out.println("Для умножения вектора-столбца на матрицу, матрица

должна состоять из 1 строки.");

else

System.out.println("Для умножения вектора-строки на матрицу, число

столбцов вектора должно совпадать с числом строк

матрицы.");

System.out.println();

return vectorType;

}

private static void writeCondition() {

System.out.println("Данная программа находит произведение вектора на

матрицу.");

System.out.println("Элементы вектора и матрицы - целые числа от -1000 до

1000.");

}

public static void main(String[] args) {

final Scanner scan = new Scanner(System.in);

int choice = 0;

int vectorType = 0;

writeCondition();

System.out.println("Если Вы хотите вводить данные в консоль, введите 0. Если

использовать файл, введите 1.");

choice = checkChoiceInput(scan);

vectorType = chooseVectorType(scan);

if (choice == 0)

inputFromConsole(vectorType, scan);

else

inputFromFile(vectorType, scan);

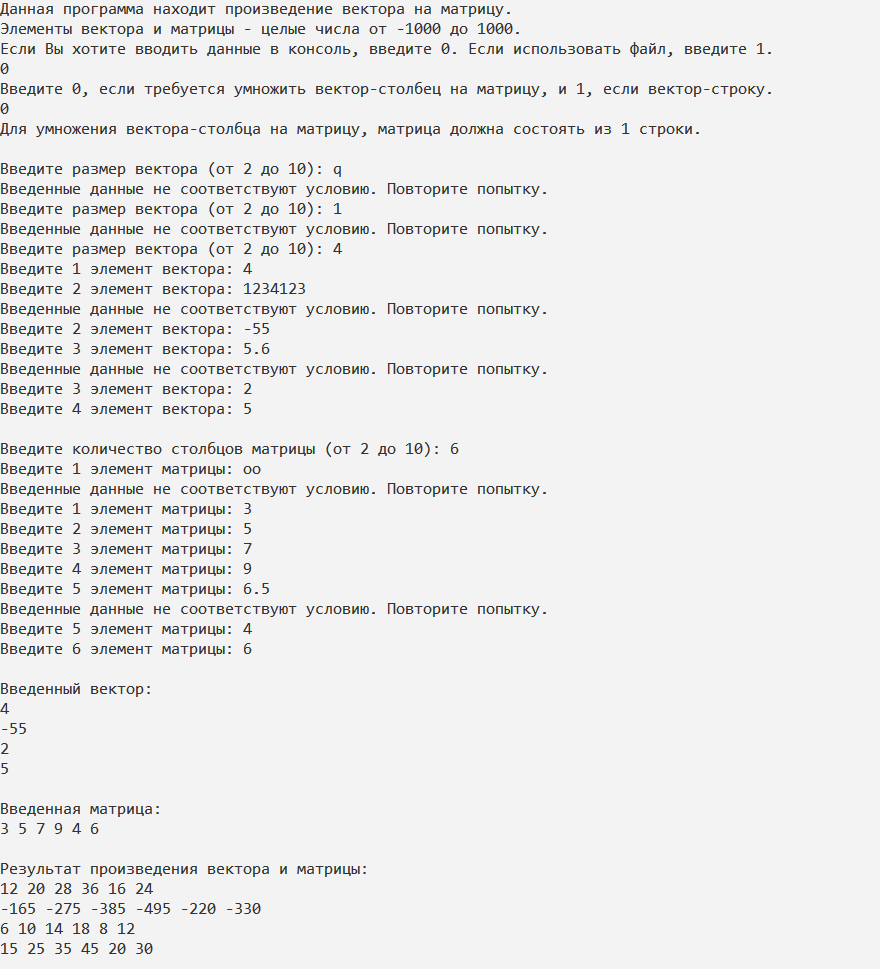
scan.close();

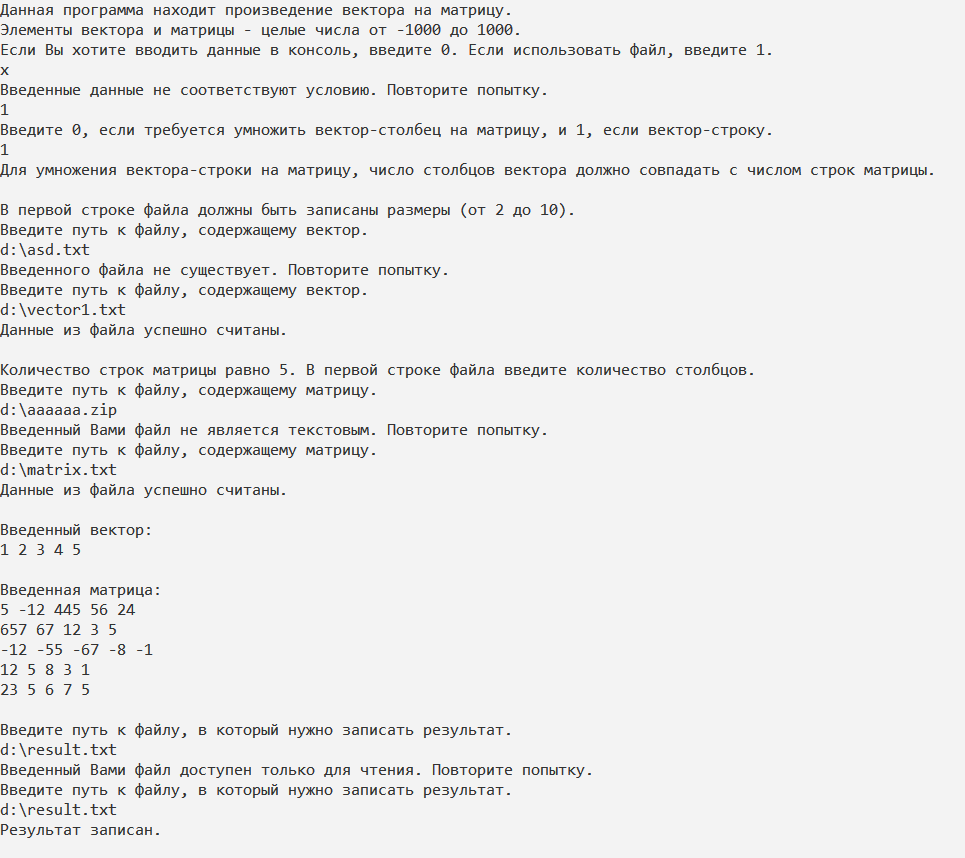
}

}

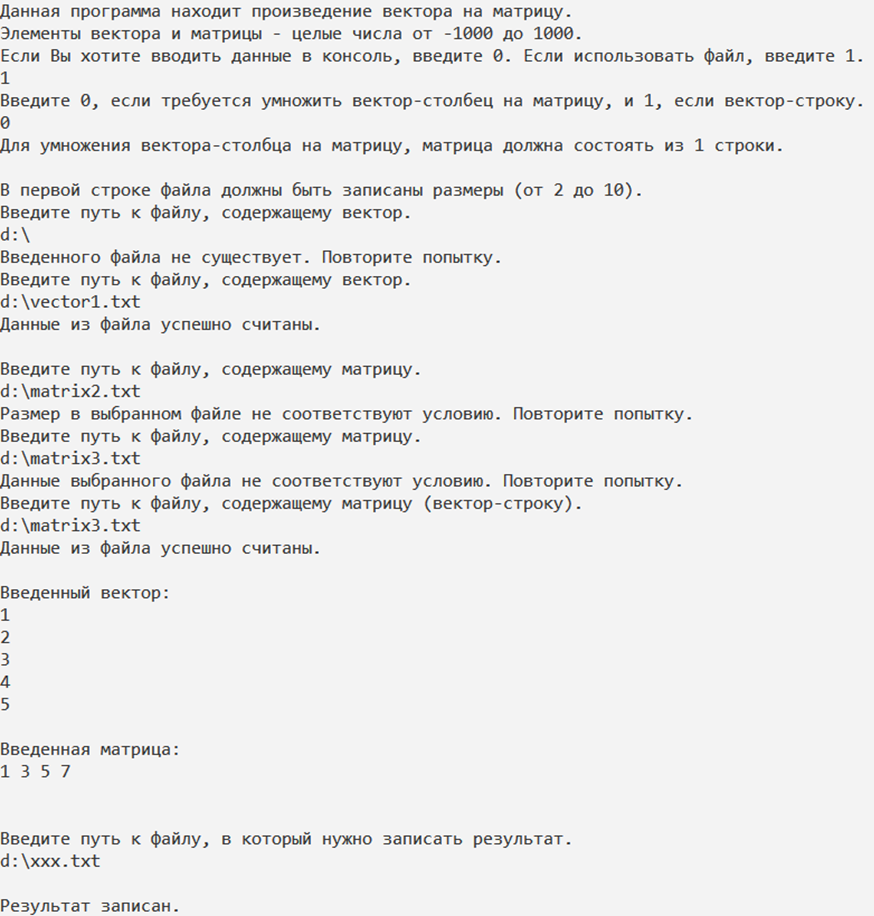
**СКРИНШОТЫ:**

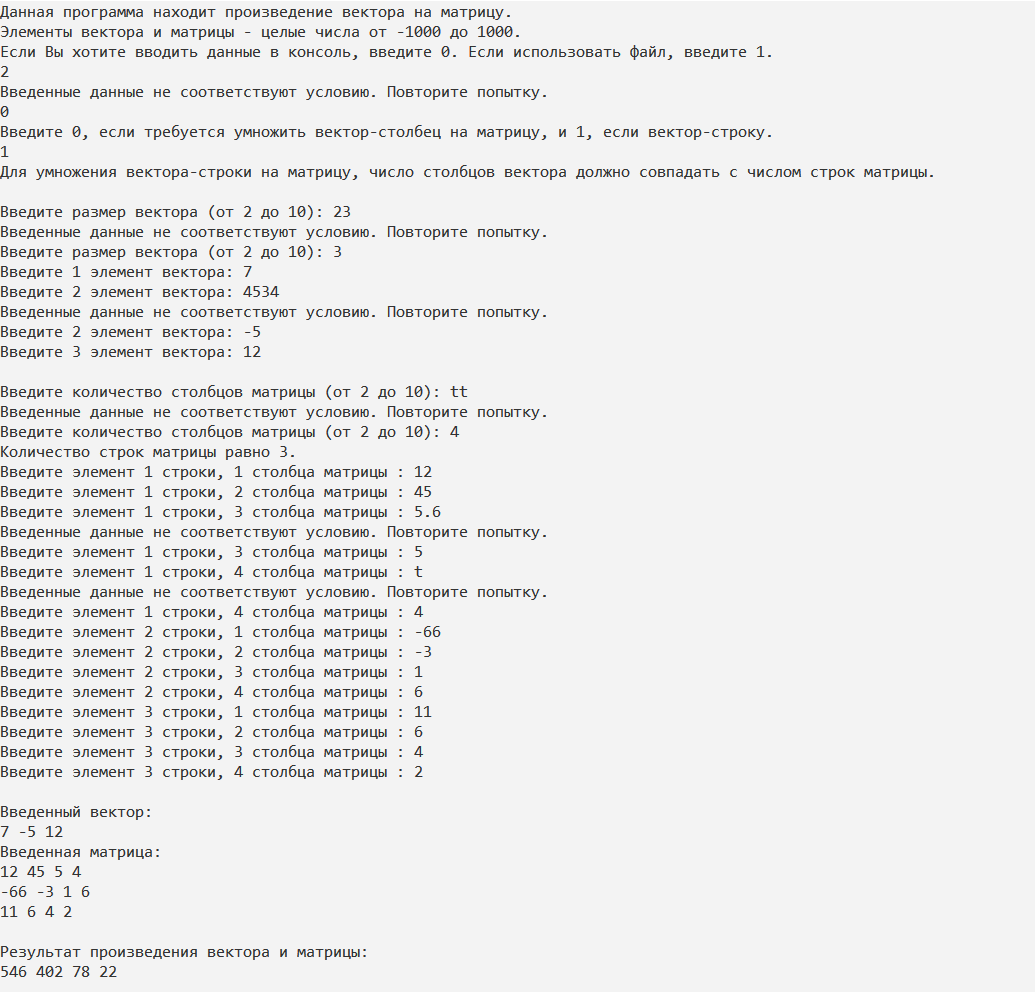
**Delphi:**



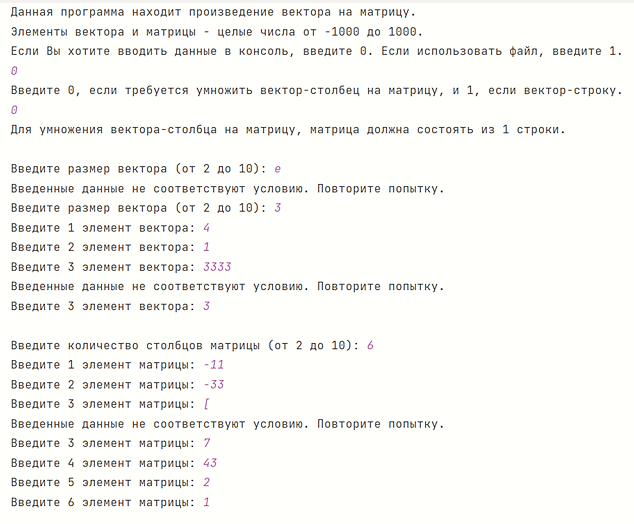


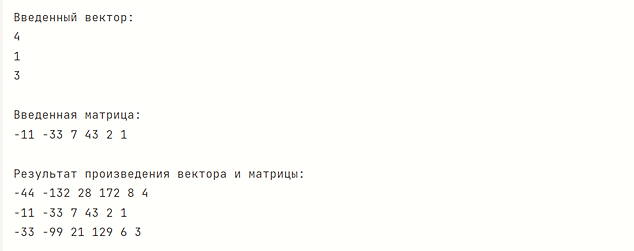
**C++:**

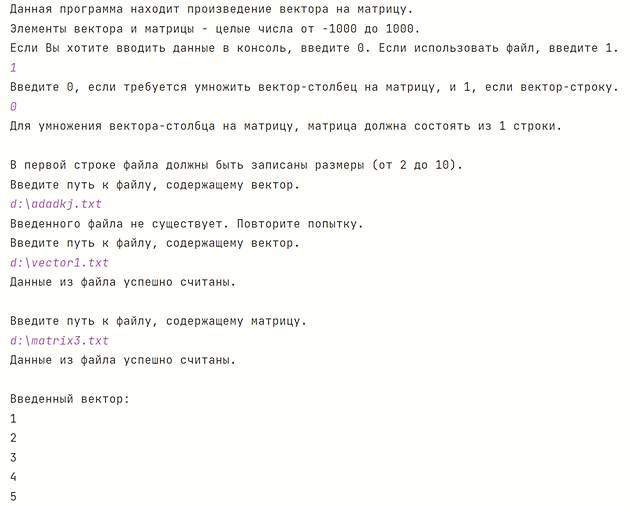
****

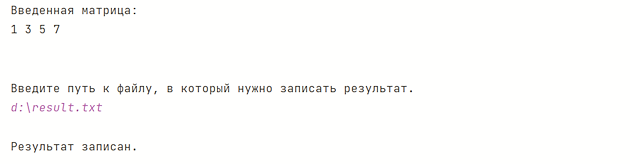


**Java:**









**БЛОК-СХЕМА:**





















