МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**Липецкий Государственный Технический Университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа

по программированию №1

“Вычисление матричных выражений”

Студент \_\_\_\_\_\_\_\_\_\_\_\_ Станиславчук С. М.

(подпись, дата)

Группа АС-21-1

Руководитель

Доцент, кандидат наук \_\_\_\_\_\_\_\_\_\_\_\_ Харитоненко А. А.

(подпись, дата)

Липецк 2022 г.

Содержание отчета

1. Титульный лист

2. Цель работы, задание

3. Описание алгоритма (блок/схема или псевдокод или диаграмма НассиШнайдермана)

4. Текст программы

5. Примеры выполнения

6. Выводы

**Цель работы**

Освоить способы динамического выделения памяти и разработки функций на примере программирования задач линейной алгебры.  
**Задание**

Вычислить заданное матричное выражение, в котором значения констант, размеры

матриц и их содержимое определяются пользователем.

Операции транспонирования, вычисления суммы, произведения на матрицу и на

константу реализовать в виде пользовательских функций. Рекомендуется также

реализовать ввод и вывод матриц в виде пользовательских функций.

Необходимо контролировать размерность матриц при осуществлении матричных

операций. В случае, если размерность матриц не позволяет провести операцию, выдавать

сообщение об ошибке.

Программа должна представлять результаты вычислений (в том числе

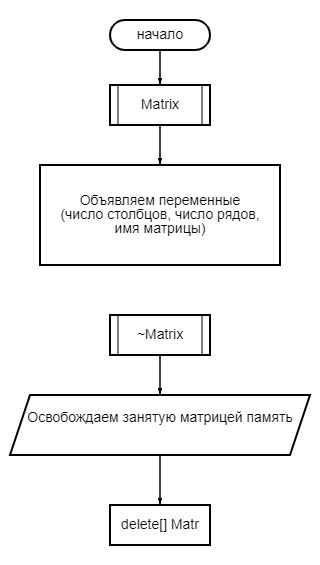
промежуточных) в виде матрицы со значениями, выровненными по столбцам, например:

4 5 5

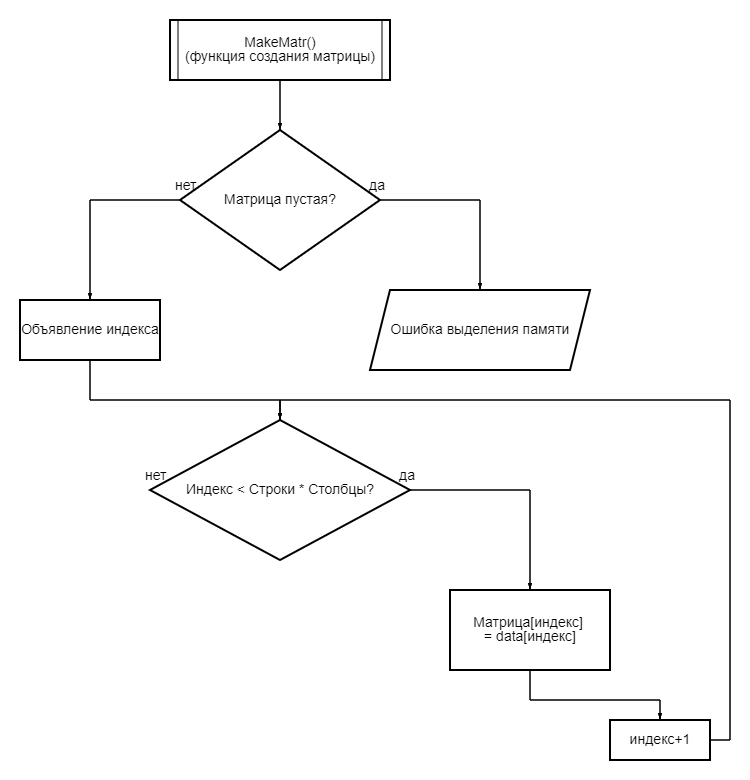
345 54 2

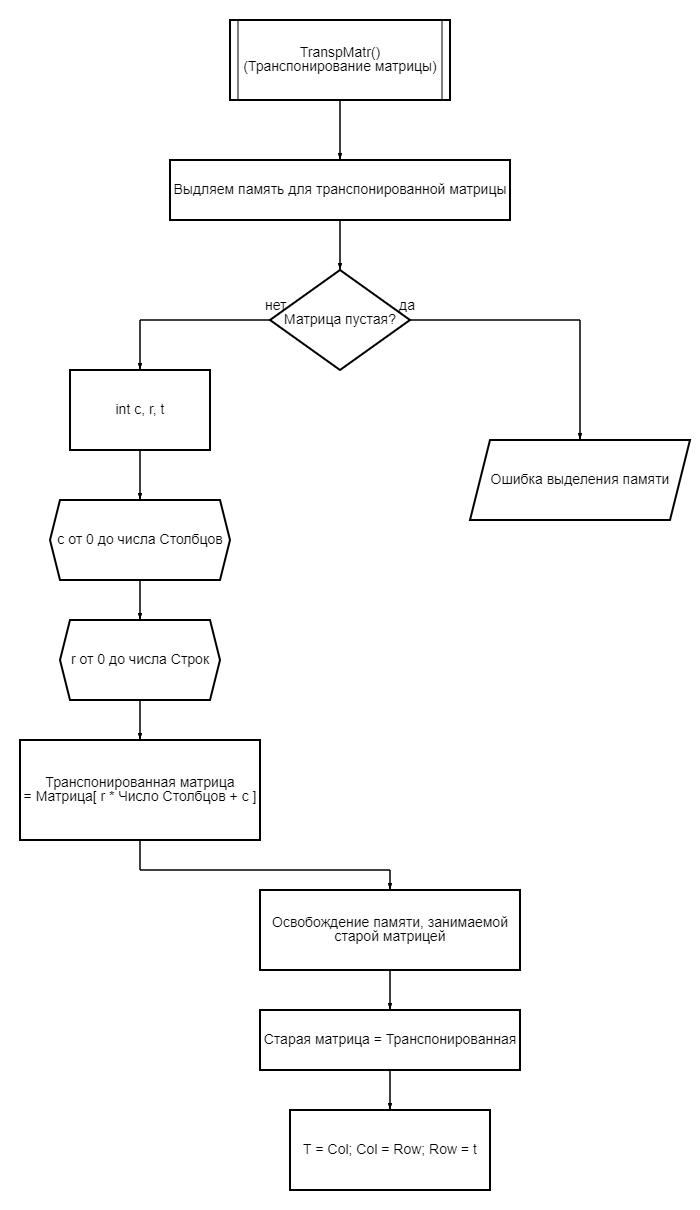
-102 4 85Вариант 1: k\*A+B\*C^T

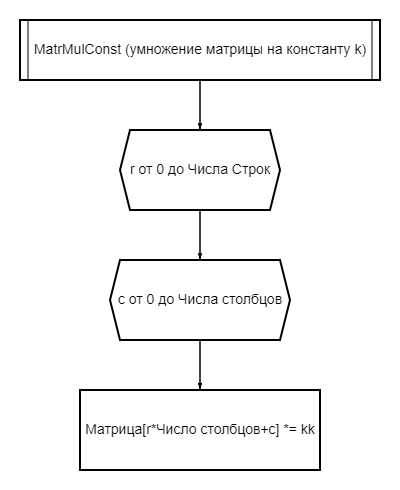
**3. Блок-схема алгоритма**

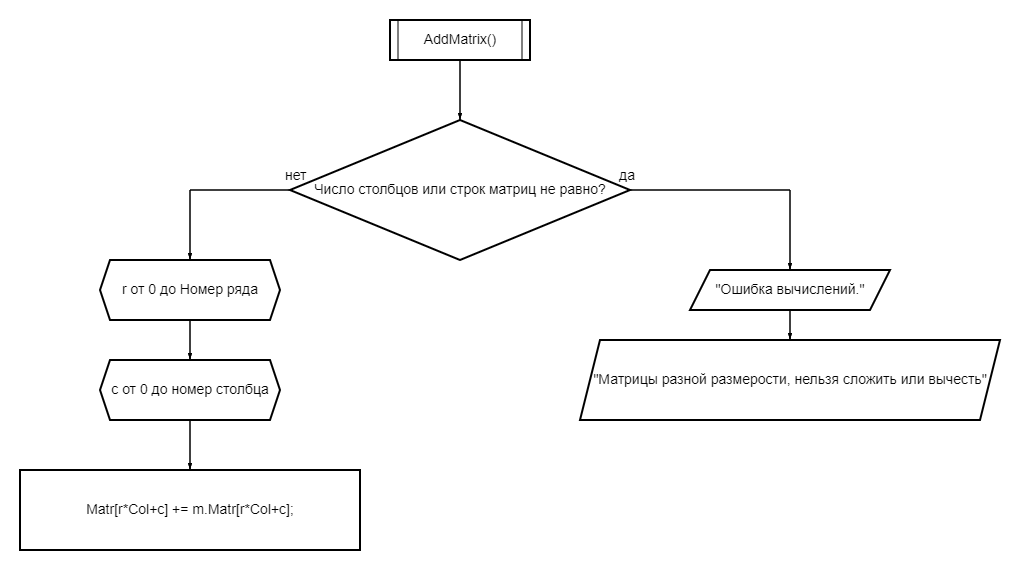


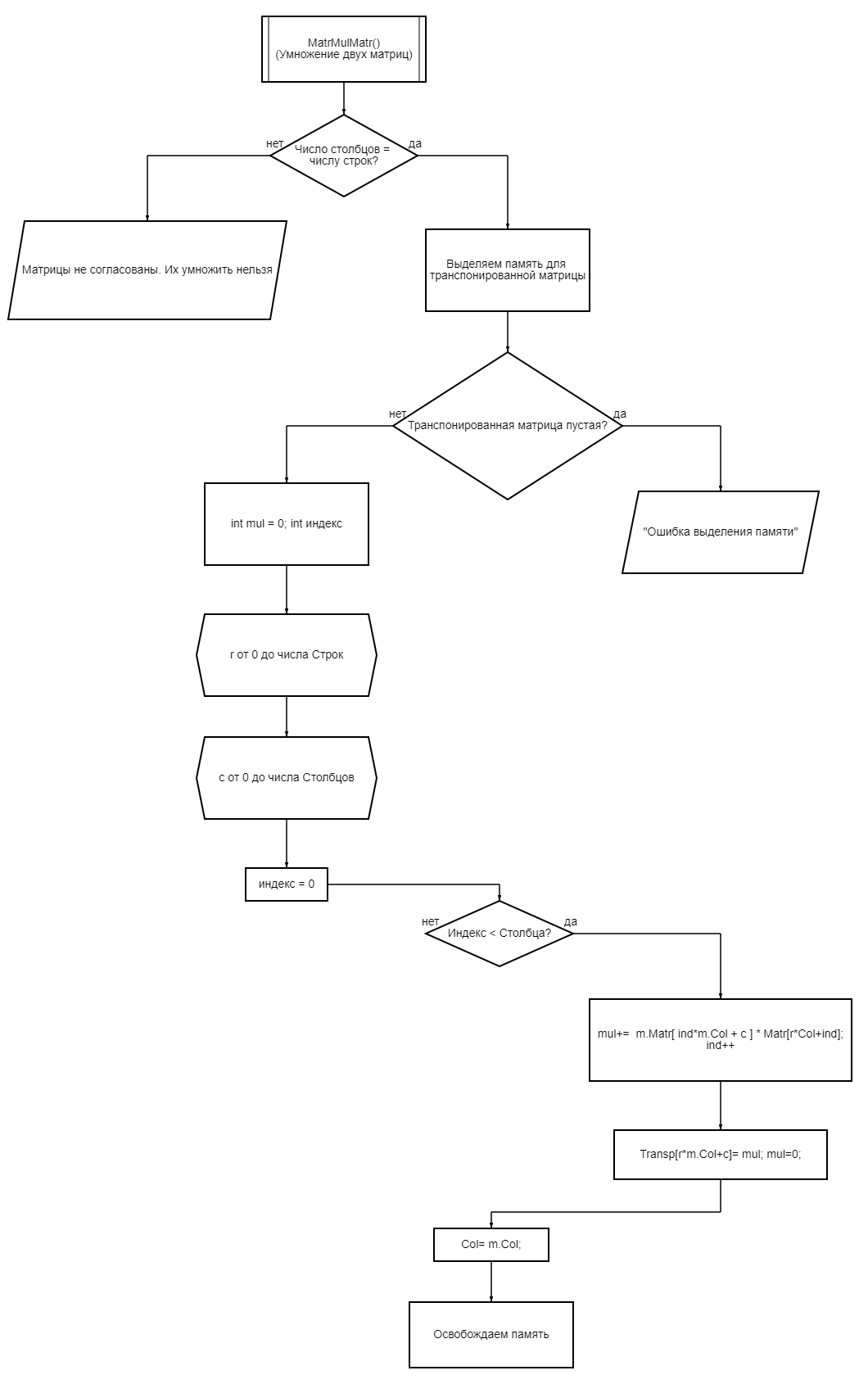


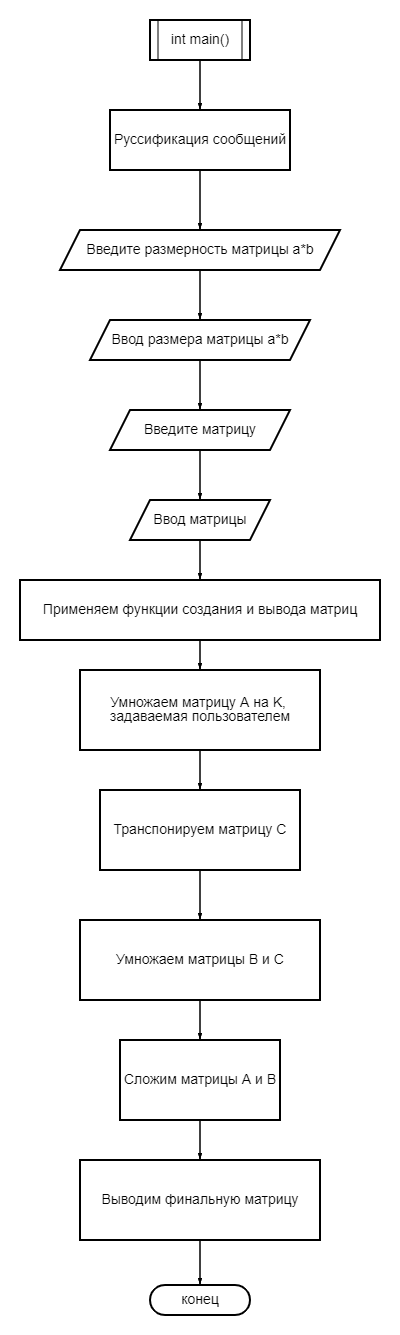












**4. Текст программы**

#include <windows.h>

#include <string>

#include <iostream>

#include <iomanip>

// k\*A+B\*C^T

using namespace std;

class Matrix

{

private:

int Col;

int Row;

public:

int \*Matr;

int \*Transp;

string NameMatr;

////////////////////

Matrix(string NM, int row, int col)

{

Col= col; Row= row; NameMatr= NM;

}

//////////////

~Matrix()

{

cout << "Освобождаем занятую матрицей " << NameMatr << " память" << endl;

delete [] Matr;

}

////////////////

void MakeMatr(int \*data)

{

Matr= new int [Row\*Col];

if (Matr==NULL) { cout << "В матрице " << NameMatr; throw 13; }

int ind=0;

while (ind < (Row\*Col) )

{

Matr[ind]= data[ind]; ind++;

}

}

////////////////

void MakeMatrB(int \*data)

{

Matr= new int [Row\*Col]; Matr= NULL;

if (Matr==NULL) { cout << "В матрице " << NameMatr; throw 13; }

int ind=0;

while (ind < (Row\*Col) )

{

Matr[ind]= data[ind]; ind++;

}

}

/////////////////

void OutMatr()

{

cout << "Матрица " << NameMatr << endl;

for (int r=0; r<Row; r++)

{

for (int c=0; c<Col; c++)

{

cout << setw(5) << Matr[r\*Col+c];

}

cout << endl;

}

cout << endl;

}

//////////////////

void TranspMatr()

{

Transp= new int [Row\*Col];

if (Transp==NULL) { cout << "В матрице " << NameMatr; throw 13; }

int c,r,t=0;

for (c=0; c<Col; c++)

{

for (r=0; r<Row; r++)

{

Transp[t]= Matr[r\*Col+c]; t++;

}

}

delete [] Matr;

Matr= Transp; Transp= NULL;

t= Col; Col= Row; Row= t;

}

///////////////

void MatrMulConst(int kk)

{

for (int r=0; r<Row; r++)

{

for (int c=0; c<Col; c++)

{

Matr[r\*Col+c] \*= kk;

}

}

}

/////////////

void MatrMulMatr(class Matrix &m)

{

if (m.Row!=Col) { throw 11; }

Transp= new int [Row\*m.Col];

if (Transp==NULL) { cout << "В матрице " << NameMatr; throw 13; }

int mul= 0; int ind;

for (int r=0; r<Row; r++)

{

for (int c=0; c< m.Col; c++)

{

ind= 0;

while (ind<Col)

{

mul+= m.Matr[ ind\*m.Col + c ] \* Matr[r\*Col+ind];

ind++;

}

Transp[r\*m.Col+c]= mul; mul=0;

}

}

Col= m.Col;

delete [] Matr;

Matr= Transp; Transp= NULL;

}

///////////////

void AddMatrix(class Matrix &m)

{

if (m.Col!=Col || m.Row!=Row) throw 12;

for (int r=0; r<Row; r++)

{

for (int c=0; c<Col; c++)

{

Matr[r\*Col+c]= Matr[r\*Col+c] + m.Matr[r\*Col+c];

}

}

}

};

int main(int argc, char \*\*argv)

{

system("chcp 1251 > nul"); // Руссификация сообщений

setlocale(LC\_ALL, "Russian");

int a, b, c;

cout << "Введите размерность матрицы A:";

cin >> a; cin >> b;

Matrix mtrA("A",a,b); int DataA[a\*b];

cout << "Введите матрицу A:";

for (int i=0; i < a\*b; i++)

cin >> DataA[i];

cout << "Введите размерность матрицы B:";

cin >> a; cin >> b;

Matrix mtrB("B",a,b); int DataB[a\*b];

cout << "Введите матрицу B:";

for (int i=0; i < a\*b; i++)

cin >> DataB[i];

cout << "Введите размерность матрицы C:";

cin >> a; cin >> b;

Matrix mtrC("C",a,b); int DataC[a\*b];

cout << "Введите матрицу C:";

for (int i=0; i< a\*b; i++)

cin >> DataC[i];

try

{

mtrA.MakeMatr(DataA); mtrB.MakeMatr(DataB); mtrC.MakeMatr(DataC);

mtrA.OutMatr(); mtrB.OutMatr(); mtrC.OutMatr();

int k;

cout << "Умножим матрицу A на k = "; cin >> k;

mtrA.MatrMulConst(k);

mtrA.OutMatr();

cout << "Транспонируем матрицу" << mtrC.NameMatr << endl;

mtrC.TranspMatr(); mtrC.OutMatr();

cout << "Умножим матрицу B на матрицу C" << endl;

mtrB.MatrMulMatr(mtrC);

mtrB.OutMatr();

cout << "Сложим матрицу A и B" << endl;

mtrA.AddMatrix(mtrA); mtrA.OutMatr();

cout << "Финальная матрица: ";

mtrA.OutMatr();

}

catch(int i)

{

switch(i)

{

case 13: cout<<" ошибка выделения памяти" << endl;

break;

case 11: cout << "Матрицы не согласованы. Их умножить нельзя" << endl;

break;

case 12: cout << "Ошибка вычислений." << endl;

cout << "Матрицы разной размерости нельзя сложить или вычесть" << endl;

break;

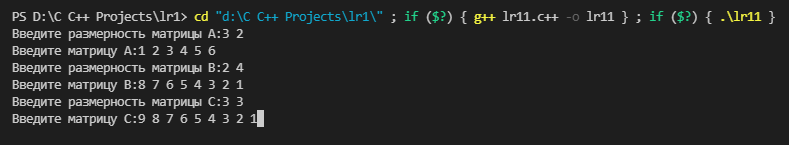
}

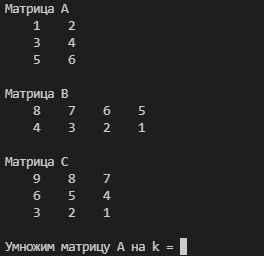
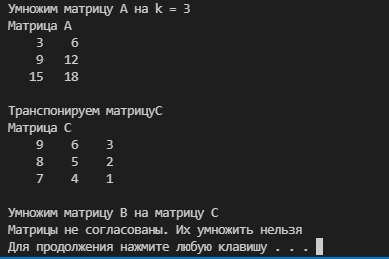
}

system("pause"); // system("pause > nul");

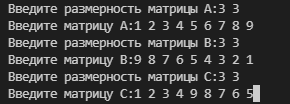
return 0;}

**5. Пример выполнения программы  
Пример №1**

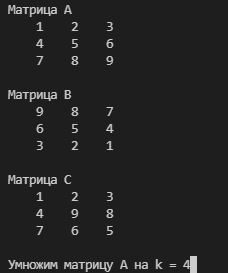


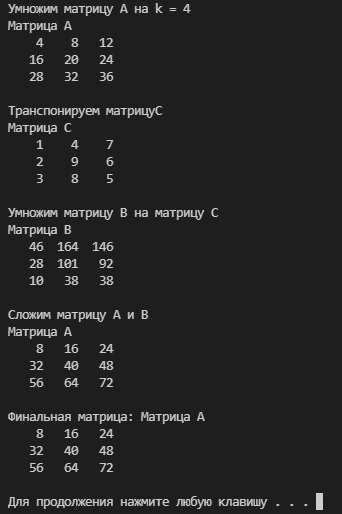
Задали параметры матриц, получили на выходе:  
  
k = 3  


**Пример №2**



K = 4



Результат:  
  
В этот раз программа выполнилась успешно.

**6. Вывод**

Освоил способы динамического выделения памяти и разработки функций на примере программирования задач линейной алгебры.