Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рязанский государственный радиотехнический университет
имени В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

Отчет
по лабораторной работе № 2

по дисциплине

«Объектно-ориентированное программирование»

на тему

«Простые классы»

Выполнил:
студент гр. 143
Вербицкая И. С.

Проверил:
Антипов О.В.

Рязань 2022

Задание

В данной лабораторной работе необходимо преобразовать лабораторную работу №1 таким образом, чтобы заданная сущность была представлена в виде класса. Класс должен быть обязательно оформлен отдельными файлами описания (.h) и реализации (.cpp). Данный класс должен иметь конструктор с параметрами, необходимыми для создания сущности, методы для реализации всех нужных операций над сущностью. Доступ к полям класса должен быть приватным, извне доступ к ним осуществляется исключительно через геттеры и сеттеры класса. Например, для структуры Person из предыдущей лабораторной

Также в данном задании необходимо написать класс юнит-тестов для каждого нетривиального метода созданного класса (см. ниже «Юнит-тестирование»). **Количество тестов должно быть не менее 10.** Результаты работы юнит-тестов необходимо выводить в консоль. Функция main должна содержать только запуск всех юнит-тестов.

Каждый тест должен быть написан по принципу **arrange -> act -> assert** и осуществлять вывод в консоль следующей информации:

- Название теста (какая функция тестируется)
- Исходные данные
- Ожидаемый результат операции
- Фактический результат операции
- Тест пройден \ Тест провален

В конце класс юнит-тестов должен вывести количество пройдённых и проваленных тестов.

Описание структуры программы

Класс Matrix3x3

Поля класса:

- **arr** - двумерный массив действительных чисел размерностью 3 на 3;
- **id** - логическое значение; 1 – если матрица единичная; 0 – если нет.

Методы класса:

- **bool Ident () -** функция по определению поля id в структуре матрицы; возвращает 0 если матрица не единичная и 1 – если единичная;
- **Matrix3x3 () –** конструктор класса; заполняет матрицу нулями;
- **void Create (Matr[N][N]) –** заполняет матрицу объекта класса элементами матрицы Matr[N][N] и переопределяет поле id для новых значений;
- **void Summ (Matrix3x3 A, Matrix3x3 B) –** записывает в поля структуры новую матрицу – результат сложения матриц A и B;
- **void Mult (Matrix3x3 A, Matrix3x3 B)** – записывает в поля структуры новую матрицу – результат перемножения матриц A и B;
- **void Scalar (Matrix3x3 int n)** – умножает каждый элемент матрицы на скаляр;
- **void Transpos () –** транспонирует матрицу;
- **float Det () –** находит и возвращает определитель матрицы;
- **bool Reverse () -** преобразует матрицу в обратную; если преобразование невозможно – возвращает 1, иначе – 0;
- **void Out (int n) –** выводит матрицу в консоль, где n – её условный номер (название);
- **float Determ (float A[N][N],int n) –** вспомогательная функция; находит определитель матрицы, представленной в виде двумерного массива A размерности n;
- **bool Compare (float Matr[N][N])** – сравнивает поля матрицы с матрицей Matr, переданной как двумерный массив.

Класс Test

Поля класса:

- Pass – целочисленное значение – количество успешно пройденных тестов
- Numb – целочисленный счётчик количества тестов в целом

Методы класса:

- void Start () – обнуляет счётчики в полях класса
- bool Expect (int Actual, int Expected) – сравнивает значения Actual и Expected между собой и возвращает 1, если они совпадают
- bool ExpectM (Matrix3x3 Actual, float Expected[N][N], int n) сравнивает объект - матрицу Actual – с двумерным массивом - матрицей Expected - и возвращает 1, если они совпадают
- void Success (bool n) – выводит сообщение об успехе/неудаче теста
- void Results () – выводит результаты теста
- void Run () – запускает Unit-тестирование

Все нижеописанные методы возвращают 1 в случае совпадения полученного результата с ожидаемым и 0 – в противном случае.

- bool TSumm () – тестирование метода Summ класса Matrix3x3;
- bool TMult () - тестирование метода Mult класса Matrix3x3;
- bool TSclr () - тестирование метода Scalar класса Matrix3x3;
- bool TTransp () - тестирование метода Transpos класса Matrix3x3;
- bool TDet () – тестирование метода Det класса Matrix3x3;
- bool TRev () - тестирование метода Reverse класса Matrix3x3;
- bool TMath () – тестирование методов Reverse и Mult на основе соблюдения формулы линейной алгебры $A * A^{-1} = E$
- bool TDetNull () - тестирование метода Det класса Matrix3x3 (нахождение определителя пропорциональной матрицы);
- bool TSummTranspSclr () - тестирование методов Summ, Transpos, Scalar класса Matrix3x3;
- bool TRevNever () - тестирование метода Reverse класса Matrix3x3 (попытка обращения матрицы с нулевым определителем);

Листинг программы:

Файл m2.h

```
#pragma once
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>

#define N 3 //размерность матриц
#define eps 1e-3 //погрешность при сравнении вещественных чисел

class Matrix3x3
{
private:
    //поля
    float arr[N][N];
    bool id;
    //определение поля id
    bool Ident();
    //вспомогательная функция
    //находит определитель двумерного массива
    float Determ (float A[N][N],int n);

public:
    //конструктор
    Matrix3x3() ;
    //заполнение матрицы
    void Create (float Matr[N][N]) ;
    //сложение матриц
    void Summ (Matrix3x3 A, Matrix3x3 B) ;
    //умножение матриц
    void Mult (Matrix3x3 A, Matrix3x3 B);
    //умножение на скаляр
    void Scalar (int n) ;
    //транспонирование
    void Transpos () ;
    //нахождение определителя
    float Det () ;
    //преобразование в обратную
    bool Reverse () ;
    //вывад одной матрицы
    void Out (int n) ;
    //сравнение двух матриц
    bool Compare(float Matr[N][N]);
};
```

Файл m2.cpp

```cpp
#include "m2.h"
    //определение, является ли матрица единичной
    bool Matrix3x3::Ident() {
        int i,j;
        //проверка главной диагонали
        for (i=0;i<N;i++)
            if (fabs(arr[i][i]-1)>=eps) return 0;
        //проверка остальных элементов
        for (i=0;i<N;i++)
        for (j=0;j<N;j++)
            if ((fabs(arr[i][j])>=eps)&&(i!=j)) return 0;
        return 1;
    }
    //вспомогательная функция
    //находит определитель двумерного массива
    float Matrix3x3::Determ (float A[N][N],int n) {
        int i,j,k;
        float a,b;
        float D;
        float B[N][N];//вспомогательная матрица, вычисляя определитель
                      //которой будем получать текущее алгебраическое дополнение
        //если матрица 2x2 определитель находится элементарно
        if (n==2)
            D=A[0][0]*A[1][1]-A[0][1]*A[1][0];
        //если НЕ 2x2 матрица раскладывается по первой строке, разложение происходит
        //до тех пор, пока все матрицы, определители которых ищем, не примут вид 2x2
        else {
            D=0;
            //собственно разложение по первой строке
            for (k=0;k<n;k++) {
                //заполнение текущей вспомогательной матрицы
                for (j=0;j<n-1;j++)
                for (i=0;i<n-1;i++) {
                    if (j<k) B[i][j]=A[i+1][j];
                    else B[i][j]=A[i+1][j+1];
                }
                a=A[0][k]; //элемент первой строки, положение которого зависит от k
                b=Determ(B,n-1); //определитель вспомогательной матрицы
                 //определение знака алгебраического дополнения
                if ((2+k)%2==0) D=D+a*b;
                else D=D-a*b;
            }
        }
        return D;
    }
    //конструктор
    Matrix3x3::Matrix3x3() {
        int i,j;
        for (i=0;i<N;i++)
        for (j=0;j<N;j++)
            arr[i][j]=0;
        id=Ident();
    }
    //заполнение матрицы
    void Matrix3x3::Create (float Matr[N][N]) {
        int i,j;
        for (i=0;i<N;i++)
        for (j=0;j<N;j++) {
            //printf("[%d][%d] = ",i+1,j+1);
            //scanf("%f",&A[i][j]);
            arr[i][j]=Matr[i][j];
        }
        id=Ident();
    }
```

```
65      //сложение матриц
66      void Matrix3x3::Summ (Matrix3x3 A, Matrix3x3 B) {
67          int i,j;
68          for (i=0;i<N;i++)
69          for (j=0;j<N;j++)
70              this->arr[i][j]=A.arr[i][j]+B.arr[i][j];
71          this->id=this->Ident();
72      }
73      //умножение матриц
74      void Matrix3x3::Mult (Matrix3x3 A, Matrix3x3 B) {
75          int i,j,k;
76          float S; //для накопления текущей суммы текущего элемента матрицы C
77          for (i=0;i<N;i++)
78          for (j=0;j<N;j++) {
79              S=0;
80              for (k=0;k<N;k++)
81                  S+=A.arr[i][k]*B.arr[k][j];
82              this->arr[i][j]=S;
83          }
84          this->id=this->Ident();
85      }
86      //умножение на скаляр
87      void Matrix3x3::Scalar (int n) {
88          int i,j;
89          for (i=0;i<N;i++)
90          for (j=0;j<N;j++)
91              arr[i][j]*=n;
92          id = Ident();
93      }
94      //транспонирование
95      void Matrix3x3::Transpos () {
96          int i,j;
97          float A[N][N]; //дополнительный массив
98          for (i=0;i<N;i++)
99          for (j=0;j<N;j++)
100             A[i][j]=arr[i][j];
101         for (i=0;i<N;i++)
102         for (j=0;j<N;j++)
103             arr[i][j]=A[j][i];
104         id=Ident();
105     }
106     //нахождение определителя
107     float Matrix3x3::Det () {
108         int i,j;
109         float D;
110         float A[N][N]; //вспомогательная матрица
111         //переносим элементы структуры во вспомогательную матрицу
112         for (i=0;i<N;i++)
113         for (j=0;j<N;j++)
114             A[i][j]=arr[i][j];
115         D=Determ(A,N); //считаем её определитель
116         return D;
117     }
```

```
118     //преобразование в обратную
119     bool Matrix3x3::Reverse () {
120         int i,j,k,q;
121         float A[N][N]; //вспомогательная матрица
122         //переносим элементы структуры во вспомогательную матрицу
123         for (i=0;i<N;i++)
124         for (j=0;j<N;j++)
125             A[i][j]=arr[i][j];
126         float B[N][N]; //присоединенная матрица из алгебраических дополнений
127         float C[N][N]; //матрица для нахождения текущего минора
128         //проходимся по всем элементам исходной матрицы для каждого - свой минор,
129         //поэтому заполняем C в зависимости от положения текущего элемента
130         for (k=0;k<N;k++)
131         for (q=0;q<N;q++) {
132             //собственно заполнение C
133             for (i=0;i<N;i++)//
134             for (j=0;j<N;j++) {
135                 if (i<q) {
136                     if (j<k) C[i][j]=A[i][j];
137                     else if (j>k) C[i][j-1]=A[i][j];
138                 }
139                 else if (i>q) {
140                     if (j<k) C[i-1][j]=A[i][j];
141                     else if (j>k) C[i-1][j-1]=A[i][j];
142                 }
143             }
144             //считаем минор текущей матрицы, составленной для текущего элемента A[i][j]
145             //тк присоединённая матрица должна быть транспонированной - [q][k], а не [k][q]
146             B[q][k]=Determ(C,N-1);
147             //определяем знак, т.е. вносим алгебраическое дополнение в присоединённую матрицу B
148             if ((k+q+2)%2!=0) B[q][k]*=-1;
149         }
150 
151         if (Determ(A,N)==0) return 1; //случай попытки найти обратную матрицу для матрицы с нулевым определителем
152         else {
153             //коэффициент, на который домножаем присоединенную матрицу для получения обратной
154             float D=1/Determ(A,N);
155             //собственно домножаем
156         for (i=0;i<N;i++)
157         for (j=0;j<N;j++)
158             B[i][j]*=D;
159         //вносим полученную матрицу обратно в структуру
160         for (i=0;i<N;i++)
161         for (j=0;j<N;j++)
162             arr[i][j]=B[j][i];
163         id=Ident();
164         return 0;
165         }
166     }
167     //вывод одной матрицы
168     void Matrix3x3::Out (int n) {
169         int i,j;
170         printf("\tМАТРИЦА %d\n",n);
171         for (i=0;i<N;i++) {
172             for (j=0;j<N;j++) printf("%*.3f  ", 7 , arr[i][j]);
173             printf("\n");
174         }
175         if (id==0) printf("\t     НЕ единичная\n");
176         else printf("\t        Единичная\n");
177     }
178     //сравнение с другой матрицей
179     bool Matrix3x3::Compare(float Matr[N][N]) {
180         int i,j;
181         for (i=0;i<N;i++)
182         for (j=0;j<N;j++)
183             if (fabs(Matr[i][j]-arr[i][j])>=eps) return 0;
184         return 1;
```

Файл t1.h

```
#pragma once
#include "m2.h"

//класс теста:
class Test
{
    int Pass; //количество пройденных тестов
    int Numb; //общее количество тестов

    //обнуление счетчиков
    void Start();
    //сравнение результатов
    //для величин
    bool Expect(int Actual, int Expected);
    //для матриц
    bool ExpectM(Matrix3x3 Actual, float Expected[N][N], int n);
    //вывод сообщения об успехе/неудаче
    void Success(bool n);
    //вывод результатов теста
    void Results();

    //возможные тесты:
    bool TSumm();
    bool TMult();
    bool TSclr();
    bool TTransp();
    bool TDet();
    bool TRev();
    bool TMath();
    bool TDetNull();
    bool TSummTranspSclr();
    bool TRevNever();

public:
    //запуск Unit-тестирования
    void Run();
};
```

Файл t1.cpp

```cpp
#include "t1.h"
using namespace std;

//результаты и исходные данные тестов:

float Test1M1[N][N]={3,9,10,-11,23,0,4,7,9};
float Test1M2[N][N]={-9,3,11,5,7,8,4,0,55};
//сумма 1 и 2
float Test1R[N][N]={-6,12,21,-6,30,8,8,7,64};

float Test2M1[N][N]={3,9,10,-11,23,0,4,7,9};
float Test2M2[N][N]={-9,3,11,5,7,8,4,0,55};
//произведение 1 и 2
float Test2R[N][N]={58,72,655,214,128,63,35,61,595};

float Test3M[N][N]={-9,3,11,5,7,8,4,0,55};
int Test3Sclr=2;
//умножение на Sclr
float Test3R[N][N]={-18,6,22,10,14,16,8,0,110};

float Test4M[N][N]={3,9,10,-11,23,0,4,7,9};
//транспонирование
float Test4R[N][N]={3,-11,4,9,23,7,10,0,9};

float Test5M[N][N]={3,-11,4,9,23,7,10,0,9};
//определитель
float Test5R=-178;

float Test6M[N][N]={58,72,655,214,128,63,35,61,595};
//преобразование в обратную
float Test6R[N][N]={0.0902,-0.0036,-0.0989,-0.1561,0.0145,0.1704,0.0107,-0.0013,-0.0099};

//единичная матрица
float Test7R[N][N]={1,0,0,0,1,0,0,0,1};

//пропорциональная матрица
float Test8M[N][N]={1,2,3,4,5,6,7,8,9};

//сумма T1M1 и T1M2 + транспонирование результата +умножение на Sclr
float Test9R[N][N]={18,18,-24,-36,-90,-21,-63,-24,-192};
int Test9Sclr=-3;


//обнуление счётчиков
void Test::Start() {
    Pass=0;
Numb=0;
}
//сравнение результатов
//для величин
bool Test::Expect(int Actual, int Expected) {
    cout << "Ожидаемый результат: " << Expected << endl;
    cout << "Фактический результат: " << Actual << endl;
    return Actual == Expected;
}
//для матриц
bool Test::ExpectM(Matrix3x3 Actual, float Expected[N][N], int n) {
    cout << "Ожидаемая матрица № "<< n << ":" << endl;
    int i,j;
    bool I; //проверка на единичность двумерного массива
    I=1;
    for (i=0;i<N;i++) {
        for (j=0;j<N;j++) {
            printf("%*.3f  ", 7 , Expected[i][j]);
            if ((fabs(Expected[i][i]-1)>=eps||((fabs(Expected[i][j])>=eps)&&(i!=j))))
                I=0;
        }
        printf("\n");
    }
    if (!I) printf("\t     НЕ единичная");
    else printf("\t        Единичная");
    cout << "\n\nФактический результат: " << endl;
    Actual.Out(n);
    return Actual.Compare(Expected);
}
//вывод сообщения об успехе/неудаче
void Test::Success(bool n) {
    ++Numb;
    if (n) ++Pass, cout << "\nУспех!" << endl;
    else cout << "\nНеудача!" << endl;
}
```

```
82  //вывод результатов теста
83  void Test::Results()
84  {
85      cout << "\n\nРЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ КЛАССА MATRI3X3" << endl;
86      cout << "Тестов пройдено: " << Pass << " из " << Numb << endl;
87  }
88
89  //сами тесты
90  bool Test::TSumm() {
91      //arrange
92      cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
93      Matrix3x3 A,B,C;
94      A.Create(Test1M1); A.Out(1);
95      B.Create(Test1M2); B.Out(2);
96      cout << "\nСЛОЖЕНИЕ МАТРИЦ 1 и 2\n" << endl;
97      //act
98      C.Summ(A,B);
99      //assert
100     return ExpectM(C,Test1R,3);
101 }
102 bool Test::TMult() {
103     //arrange
104     cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
105     Matrix3x3 A,B,C;
106     A.Create(Test2M1); A.Out(1);
107     B.Create(Test2M2); B.Out(2);
108     cout << "\nУМНОЖЕНИЕ МАТРИЦ 1 и 2\n" << endl;
109     //act
110     C.Mult(A,B);
111     //assert
112     return ExpectM(C,Test2R,3);
113 }
114 bool Test::TSclr() {
115     //arrange
116     cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
117     Matrix3x3 A;
118     A.Create(Test3M);
119     A.Out(1);
120     cout << "\nУМНОЖЕНИЕ МАТРИЦЫ 1 НА ЧИСЛО " << Test3Sclr << "\n" << endl;
121     //act
122     A.Scalar(Test3Sclr);
123     //assert
124     return ExpectM(A,Test3R,1);
125 }
126 bool Test::TTransp() {
127     //arrange
128     cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
129     Matrix3x3 A;
130     A.Create(Test4M);
131     A.Out(1);
132     cout << "\nТРАНСПОНИРОВАНИЕ МАТРИЦЫ 1\n" << endl;
133     //act
134     A.Transpos();
135     //assert
136     return ExpectM(A,Test4R,1);
137 }
138 bool Test::TDet() {
139     //arrange
140     cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
141     Matrix3x3 A;
142     A.Create(Test5M);
143     A.Out(1);
144     cout << "\nНАХОЖДЕНИЕ ОПРЕДЕЛИТЕЛЯ МАТРИЦЫ 1\n" << endl;
145     //act
146     float D;
147     D=A.Det();
148     //assert
149     return Expect(D,Test5R);
150 }
151 bool Test::TRev() {
152     //arrange
153     cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
154     Matrix3x3 A;
155     A.Create(Test6M);
156     A.Out(1);
157     cout << "\nПРЕОБРАЗОВАНИЕ МАТРИЦЫ 1 В ОБРАТНУЮ\n" << endl;
158     //act
159     A.Reverse();
160     //assert
161     return ExpectM(A,Test6R,1);
162 }
```

```cpp
bool Test::TMath() {
    //arrange
    cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
    Matrix3x3 A,B,C;
    A.Create(Test6M); B.Create(Test6M);
    A.Out(1); B.Out(2);
    cout << "\nПРЕОБРАЗОВАНИЕ МАТРИЦЫ 1 В ОБРАТНУЮ, А ЗАТЕМ ПЕРЕМНОЖЕНИЕ МАТРИЦ 1 И 2" << endl;
    cout << "(ПРОВЕРКА ФОРМУЛЫ ИЗ ЛИНЕЙНОЙ АЛГЕБРЫ A*A^(-1)=E)\n" << endl;
    //act
    A.Reverse();
    C.Mult(A,B);
    //assert
    return ExpectM(C,Test7R,3);
}
bool Test::TDetNull() {
    //arrange
    cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
    Matrix3x3 A;
    A.Create(Test8M);
    A.Out(1);
    cout << "\nНАХОЖДЕНИЕ ОПРЕДЕЛИТЕЛЯ ПРОПОРЦИОНАЛЬНОЙ МАТРИЦЫ 1\n" << endl;
    //act
    float D;
    D=A.Det();
    //assert
    return Expect(D,0);
}
bool Test::TSummTranspSclr() {
    //arrange
    cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
    Matrix3x3 A,B,C;
    A.Create(Test1M1); A.Out(1);
    B.Create(Test1M2); B.Out(2);
    cout << "\nСЛОЖЕНИЕ МАТРИЦ 1 и 2, А ЗАТЕМ ТРАНСПОНИРОВАНИЕ РЕЗУЛЬТАТА И УМНОЖЕНИЕ НА ЧИСЛО " << Test9Sclr << "\n"<< endl;
    //act
    C.Summ(A,B);
    C.Transpos();
    C.Scalar(Test9Sclr);
    //assert
    return ExpectM(C,Test9R,3);
}
bool Test::TRevNever() {
    //arrange
    cout << "\nТЕСТ №" << Pass+1 << "\n" << endl;
    Matrix3x3 A;
    A.Create(Test8M);
    A.Out(1);
    cout << "\nПРЕОБРАЗОВАНИЕ МАТРИЦЫ 1 В ОБРАТНУЮ" << endl;
    cout << "(ПОПЫТКА ПРЕОБРАЗОВАНИЯ МАТРИЦЫ С НУЛЕВЫМ ОПРЕДЕЛИТЕЛЕМ В ОБРАТНУЮ)\n" << endl;
    //act
    A.Reverse();
    //assert
    return ExpectM(A,Test8M,1);
}

//запуск Unit-тестирования
void Test::Run() {
    Start();

    Success(TSumm());
    Success(TMult());
    Success(TSclr());
    Success(TTransp());
    Success(TDet());
    Success(TRev());
    Success(TMath());
    Success(TDetNull());
    Success(TSummTranspSclr());
    Success(TRevNever());

    Results();
}
```

Файл main.cpp

```cpp
#include "t1.h"
int main() {
    system("chcp 1251");
    Test T;
    T.Run();
    return 0;
}
```

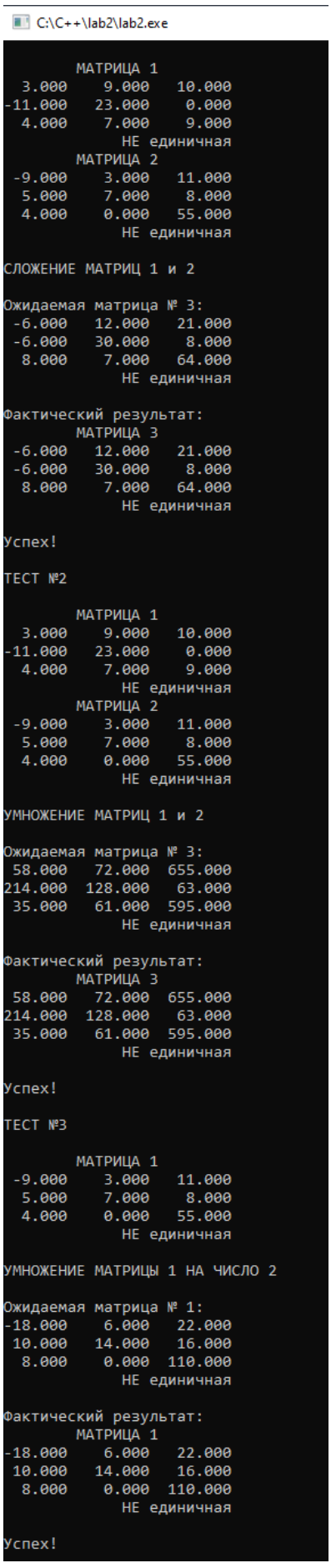

Рисунок 1

Результаты работы программы:

После запуска программа выводит на экран результаты 10-ти тестов. Большая часть из них в том или ином виде присутствовала в примерах выполнения программы и их проверках в предыдущей работе, часть же результатов была посчитана дополнительно с помощью приложения Excel. Именно эти значения и стали контрольными для выполнения тестов, именно на них опирались проверки правильности вычислений.

Итак, после запуска программа вывела в консоль результаты всех 10-ти тестов (рисунки 1, 2, 3), значения которых соответствуют действительности и совпадают с ранее выполненными проверками. Программа работает корректно.

C:\C++\lab2\lab2.exe

```
ТЕСТ №4

        МАТРИЦА 1
  3.000    9.000   10.000
-11.000   23.000    0.000
  4.000    7.000    9.000
             НЕ единичная

ТРАНСПОНИРОВАНИЕ МАТРИЦЫ 1

Ожидаемая матрица № 1:
  3.000  -11.000    4.000
  9.000   23.000    7.000
 10.000    0.000    9.000
             НЕ единичная

Фактический результат:
        МАТРИЦА 1
  3.000  -11.000    4.000
  9.000   23.000    7.000
 10.000    0.000    9.000
             НЕ единичная

Успех!

ТЕСТ №5

        МАТРИЦА 1
  3.000  -11.000    4.000
  9.000   23.000    7.000
 10.000    0.000    9.000
             НЕ единичная

НАХОЖДЕНИЕ ОПРЕДЕЛИТЕЛЯ МАТРИЦЫ 1

Ожидаемый результат: -178
Фактический результат: -178

Успех!

ТЕСТ №6

        МАТРИЦА 1
 58.000   72.000  655.000
214.000  128.000   63.000
 35.000   61.000  595.000
             НЕ единичная

ПРЕОБРАЗОВАНИЕ МАТРИЦЫ 1 В ОБРАТНУЮ

Ожидаемая матрица № 1:
  0.090   -0.004   -0.099
 -0.156    0.014    0.170
  0.011   -0.001   -0.010
             НЕ единичная

Фактический результат:
        МАТРИЦА 1
  0.090   -0.004   -0.099
 -0.156    0.014    0.170
  0.011   -0.001   -0.010
             НЕ единичная

Успех!

ТЕСТ №7

        МАТРИЦА 1
 58.000   72.000  655.000
214.000  128.000   63.000
 35.000   61.000  595.000
             НЕ единичная
        МАТРИЦА 2
 58.000   72.000  655.000
214.000  128.000   63.000
 35.000   61.000  595.000
             НЕ единичная

ПРЕОБРАЗОВАНИЕ МАТРИЦЫ 1 В ОБРАТНУЮ, А ЗАТЕМ ПЕРЕМНОЖЕНИЕ МАТРИЦ 1 И 2
(ПРОВЕРКА ФОРМУЛЫ ИЗ ЛИНЕЙНОЙ АЛГЕБРЫ A*A^(-1)=E)

Ожидаемая матрица № 3:
  1.000    0.000    0.000
  0.000    1.000    0.000
  0.000    0.000    1.000
```

Рисунок 2

```
Фактический результат:
        МАТРИЦА 3
  1.000    0.000    0.000
 -0.000    1.000   -0.000
 -0.000   -0.000    1.000
               Единичная

Успех!

ТЕСТ №8

        МАТРИЦА 1
  1.000    2.000    3.000
  4.000    5.000    6.000
  7.000    8.000    9.000
             НЕ единичная

НАХОЖДЕНИЕ ОПРЕДЕЛИТЕЛЯ ПРОПОРЦИОНАЛЬНОЙ МАТРИЦЫ 1

Ожидаемый результат: 0
Фактический результат: 0

Успех!

ТЕСТ №9

        МАТРИЦА 1
  3.000    9.000   10.000
-11.000   23.000    0.000
  4.000    7.000    9.000
             НЕ единичная
        МАТРИЦА 2
 -9.000    3.000   11.000
  5.000    7.000    8.000
  4.000    0.000   55.000
             НЕ единичная

СЛОЖЕНИЕ МАТРИЦ 1 и 2, А ЗАТЕМ ТРАНСПОНИРОВАНИЕ РЕЗУЛЬТАТА И УМНОЖЕНИЕ НА ЧИСЛО -3

Ожидаемая матрица № 3:
 18.000   18.000  -24.000
-36.000  -90.000  -21.000
-63.000  -24.000  -192.000
             НЕ единичная

Фактический результат:
        МАТРИЦА 3
 18.000   18.000  -24.000
-36.000  -90.000  -21.000
-63.000  -24.000  -192.000
             НЕ единичная

Успех!

ТЕСТ №10

        МАТРИЦА 1
  1.000    2.000    3.000
  4.000    5.000    6.000
  7.000    8.000    9.000
             НЕ единичная

ПРЕОБРАЗОВАНИЕ МАТРИЦЫ 1 В ОБРАТНУЮ
(ПОПЫТКА ПРЕОБРАЗОВАНИЯ МАТРИЦЫ С НУЛЕВЫМ ОПРЕДЕЛИТЕЛЕМ В ОБРАТНУЮ)

Ожидаемая матрица № 1:
  1.000    2.000    3.000
  4.000    5.000    6.000
  7.000    8.000    9.000
             НЕ единичная

Фактический результат:
        МАТРИЦА 1
  1.000    2.000    3.000
  4.000    5.000    6.000
  7.000    8.000    9.000
             НЕ единичная

Успех!


РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ КЛАССА MATRI3X3
Тестов пройдено: 10 из 10
```

Рисунок 3