

1. Введение

- 1) Нахождение обратной матрицы методом исключения неизвестных Гаусса. Первый шаг для нахождения обратной матрицы методом исключения неизвестных Гаусса - приписать к матрице A единичную матрицу того же порядка, отделив их вертикальной чертой. Мы получим сдвоенную матрицу $[A|E]$. Умножим обе части этой матрицы на A^{-1} , тогда получим $[E|A^{-1}]$, но и A^{-1} . Алгоритм нахождения обратной матрицы методом исключения неизвестных Гаусса 1. К матрице A приписать единичную матрицу того же порядка. 2. Полученную сдвоенную матрицу преобразовать так, чтобы в левой её части получилась единичная матрица, тогда в правой части на месте единичной матрицы автоматически получится обратная матрица. Матрица A в левой части преобразуется в единичную матрицу путём элементарных преобразований матрицы. 2. Если в процессе преобразования матрицы A в единичную матрицу в какой-либо строке или в каком-либо столбце окажутся только нули, то определитель матрицы равен нулю, и, следовательно, матрица A будет вырожденной, и она не имеет обратной матрицы. В этом случае дальнейшее нахождение обратной матрицы прекращается.

- 2) `include <iostream>`

```
void inversion(double **A, int N) double temp;
double **E = new double *[N];
for (int i = 0; i < N; i++) E[i] = new double [N];
for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) E[i][j] = 0.0;
if (i == j) E[i][j] = 1.0;
for (int k = 0; k < N; k++) temp = A[k][k];
for (int j = 0; j < N; j++) A[k][j] /= temp; E[k][j] /= temp;
for (int i = k + 1; i < N; i++) temp = A[i][k];
for (int j = 0; j < N; j++) A[i][j] -= A[k][j] * temp; E[i][j] -= E[k][j] * temp;
for (int k = N - 1; k > 0; k--) for (int i = k - 1; i >= 0; i--) temp = A[i][k];
for (int j = 0; j < N; j++) A[i][j] -= A[k][j] * temp; E[i][j] -= E[k][j] * temp;
for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) A[i][j] = E[i][j];
for (int i = 0; i < N; i++) delete [] E[i];
delete [] E;
int main() int N;
std::cout << "Enter N: "; std::cin >> N;
double **matrix = new double *[N];
for (int i = 0; i < N; i++) matrix[i] = new double [N];
```

```

for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) std::cout << "Enter matrix[" << i << "][" << j << "] = "; std::cin >> matrix[i][j];
inversion(matrix, N);
for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) std::cout << matrix[i][j] << " ";
std::cout << std::endl;
for (int i = 0; i < N; i++) delete [] matrix[i];
delete [] matrix;
std::cin.get(); return 0;

```

2. Ход работы

2.1. Код приложения

```

#include <iostream>
void inversion(double** A, int N)
{
    double temp;
    double** E = new double* [N];
    for (int i = 0; i < N; i++)
        E[i] = new double[N];
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
        {
            E[i][j] = 0.0;
            if (i == j)
                E[i][j] = 1.0;
        }
    for (int k = 0; k < N; k++)
    {
        temp = A[k][k];
        for (int j = 0; j < N; j++)
        {
            A[k][j] /= temp;
            E[k][j] /= temp;
        }
        for (int i = k + 1; i < N; i++)
        {
            temp = A[i][k];
            for (int j = 0; j < N; j++)
            {
                A[i][j] -= A[k][j] * temp;
            }
        }
    }
}

```

```

        E[i][j] -= E[k][j] * temp;
    }
}
for (int k = N - 1; k > 0; k--)
{
    for (int i = k - 1; i >= 0; i--)
    {
        temp = A[i][k];
        for (int j = 0; j < N; j++)
        {
            A[i][j] -= A[k][j] * temp;
            E[i][j] -= E[k][j] * temp;
        }
    }
}
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        A[i][j] = E[i][j];
for (int i = 0; i < N; i++)
    delete[] E[i];
delete[] E;
}
int main()
{
    int N;
    std::cout << "Enter N: ";
    std::cin >> N;
    double** matrix = new double* [N];
    for (int i = 0; i < N; i++)
        matrix[i] = new double[N];
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
        {
            std::cout << "Enter matrix[" << i << "][" << j << "] = ";
            std::cin >> matrix[i][j];
        }
    inversion(matrix, N);
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            std::cout << matrix[i][j] << " ";
        std::cout << std::endl;
    }
}

```

```

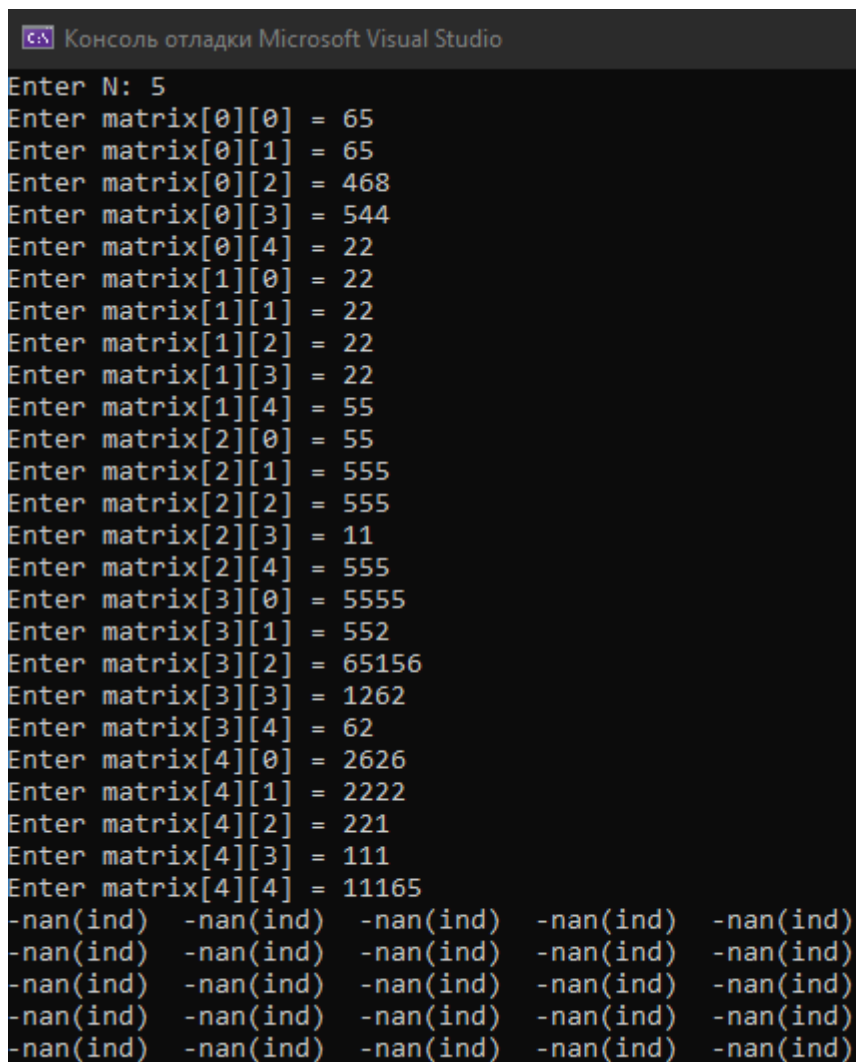
    for (int i = 0; i < N; i++)
        delete[] matrix[i];
    delete[] matrix;
    std::cin.get();
    return 0;
}

```

2.2. Формулы

$$A * (A)^{-1} = E; E * A^{-1} = A^{-1}$$

3. Скриншот



```

Консоль отладки Microsoft Visual Studio
Enter N: 5
Enter matrix[0][0] = 65
Enter matrix[0][1] = 65
Enter matrix[0][2] = 468
Enter matrix[0][3] = 544
Enter matrix[0][4] = 22
Enter matrix[1][0] = 22
Enter matrix[1][1] = 22
Enter matrix[1][2] = 22
Enter matrix[1][3] = 22
Enter matrix[1][4] = 55
Enter matrix[2][0] = 55
Enter matrix[2][1] = 555
Enter matrix[2][2] = 555
Enter matrix[2][3] = 11
Enter matrix[2][4] = 555
Enter matrix[3][0] = 5555
Enter matrix[3][1] = 552
Enter matrix[3][2] = 65156
Enter matrix[3][3] = 1262
Enter matrix[3][4] = 62
Enter matrix[4][0] = 2626
Enter matrix[4][1] = 2222
Enter matrix[4][2] = 221
Enter matrix[4][3] = 111
Enter matrix[4][4] = 11165
-nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)
-nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)
-nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)
-nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)
-nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)

```

Список литературы

- [1] Кнут Д.Э. Всё про $\text{T}_\text{E}\text{X}$. — Москва: Изд. Вильямс, 2003 г. 550 с.
- [2] Львовский С.М. Набор и верстка в системе $\text{L}_\text{A}\text{T}_\text{E}\text{X}$. — 3-е издание, исправленное и дополненное, 2003 г.
- [3] Воронцов К.В. $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ в примерах. 2005 г.