

Trabalho de Algebra Linear - Turma A

Aluno: Eduardo Larson

Faça um código que resolva uma matriz de adição, subtração, multiplicação e de Gauss-Jordan:

Matriz de adição:

```
#include <iostream>
using namespace std;
#define N 10

int main()
{
    int linha1 = 0, coluna1 = 0, linha2 = 0, coluna2 = 0;

    cout << "Adicione a quantidade de linhas e colunas da primeira
matriz: ";
    cin >> linha1 >> coluna1;

    cout << "Adicione a quantidade de linhas e colunas da segunda
matriz: ";
    cin >> linha2 >> coluna2;

    if (coluna1 != linha2)
    {
        cout << "A matriz eh Invalida.";
        return 0;
    }

    int matriz1[N][N] = {0}, matriz2[N][N] = {0}, produto[N][N] = {0};

    cout << "Adicione os elementos da primeira matriz: " << endl;
    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna1; ++j)
        {
            cin >> matriz1[i][j];
        }
    }

    cout << "Adicione os elementos da segunda matriz: " << endl;
    for (int i = 0; i < linha2; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
```

```

        {
            cin >> matriz2[i][j];
        }
    }

    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            produto[i][j] = matriz1[i][j] + matriz2[i][j];
        }
    }

    cout << "O resultado da operacao eh: " << endl;
    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            cout << produto[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Compilado:

```

Adicione a quantidade de linhas e colunas da primeira matriz: 2
2
Adicione a quantidade de linhas e colunas da segunda matriz: 2
2
Adicione os elementos da primeira matriz:
1
2
3
4
Adicione os elementos da segunda matriz:
1
2
3
4
O resultado da operacao eh:
2 4
6 8

```

Matriz de subtração:

```

#include <iostream>
using namespace std;

```

```
#define N 10

int main()
{
    int linha1 = 0, coluna1 = 0, linha2 = 0, coluna2 = 0;

    cout << "Adicione a quantidade de linhas e colunas da primeira
matriz: ";
    cin >> linha1 >> coluna1;

    cout << "Adicione a quantidade de linhas e colunas da segunda
matriz: ";
    cin >> linha2 >> coluna2;

    if (coluna1 != linha2)
    {
        cout << "A matriz eh Invalida.";
        return 0;
    }

    int matriz1[N][N] = {0}, matriz2[N][N] = {0}, produto[N][N] = {0};

    cout << "Adicione os elementos da primeira matriz: " << endl;
    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna1; ++j)
        {
            cin >> matriz1[i][j];
        }
    }

    cout << "Adicione os elementos da segunda matriz: " << endl;
    for (int i = 0; i < linha2; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            cin >> matriz2[i][j];
        }
    }

    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
```

```

        {
            produto[i][j] = matriz1[i][j] - matriz2[i][j];
        }
    }

    cout << "O resultado da operacao eh: " << endl;
    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            cout << produto[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Compilado:

```

Adicione a quantidade de linhas e colunas da primeira matriz: 2
2
Adicione a quantidade de linhas e colunas da segunda matriz: 2
2
Adicione os elementos da primeira matriz:
1
2
3
4
Adicione os elementos da segunda matriz:
1
2
3
4
O resultado da operacao eh:
0 0
0 0

```

Matriz de Multiplicação:

```

#include <iostream>
using namespace std;
#define N 10

int main()
{
    int linha1 = 0, coluna1 = 0, linha2 = 0, coluna2 = 0, sum = 0;

    cout << "Adicione a quantidade de linhas e colunas da primeira
matriz: ";

```

```
    cin >> linha1 >> coluna1;

    cout << "Adicione a quantidade de linhas e colunas da segunda
matriz: ";
    cin >> linha2 >> coluna2;

    if (coluna1 != linha2)
    {
        cout << "A matriz nao pode ser multiplicada.";
        return 0;
    }

    int matriz1[N][N] = {0}, matriz2[N][N] = {0}, produto[N][N] = {0};

    cout << "Adicione os elementos da primeira matriz: " << endl;
    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna1; ++j)
        {
            cin >> matriz1[i][j];
        }
    }

    cout << "Adicione os elementos da segunda matriz: " << endl;
    for (int i = 0; i < linha2; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            cin >> matriz2[i][j];
        }
    }

    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            for (int k = 0; k < coluna1; ++k)
            {
                produto[i][j] += matriz1[i][k] * matriz2[k][j];
            }
        }
    }
}
```

```

    cout << "O resultado da operacao eh: " << endl;
    for (int i = 0; i < linha1; ++i)
    {
        for (int j = 0; j < coluna2; ++j)
        {
            cout << produto[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}

```

Compilado:

```

Adicione a quantidade de linhas e colunas da primeira matriz: 2
2
Adicione a quantidade de linhas e colunas da segunda matriz: 2
2
Adicione os elementos da primeira matriz:
1
2
3
4
Adicione os elementos da segunda matriz:
2
3
4
5
O resultado da operacao eh:
10 13
22 29

```

Matriz Gauss-Jordan:

```

#include <iostream>
#include <vector>
#include <iomanip>

using namespace std;

void imprime(const vector<vector<double>>& matriz) {
    int linhas = matriz.size();
    int colunas = matriz[0].size();

    for (int i = 0; i < linhas; i++) {
        for (int j = 0; j < colunas; j++) {
            cout << fixed << setprecision(0) << matriz[i][j] << " ";
        }
        cout << endl;
    }
}

```

```

    }
    cout << endl;
}

void GaussJordan(vector<vector<double>>& matriz) {
    int linhas = matriz.size();
    int colunas = matriz[0].size();

    cout << "Matriz Inicial:" << endl;
    imprime(matriz);

    for (int i = 0; i < linhas; i++) {
        int pivotRow = i;
        for (int j = i + 1; j < linhas; j++) {
            if (matriz[j][i] > matriz[pivotRow][i]) {
                pivotRow = j;
            }
        }
        swap(matriz[i], matriz[pivotRow]);

        cout << "Pivoteamento Parcial (linha " << i + 1 << " trocada
com linha " << pivotRow + 1 << "):" << endl;
        imprime(matriz);

        double pivot = matriz[i][i];
        for (int j = 0; j < colunas; j++) {
            matriz[i][j] /= pivot;
        }

        cout << "Escalonamento (linha " << i + 1 << " dividida por " <<
pivot << "):" << endl;
        imprime(matriz);

        for (int j = 0; j < linhas; j++) {
            if (j != i) {
                double factor = matriz[j][i];
                for (int k = 0; k < colunas; k++) {
                    matriz[j][k] -= factor * matriz[i][k];
                }
                cout << "Eliminacao (linha " << j + 1 << " subtraida de
" << factor << " vezes a linha " << i + 1 << "):" << endl;
                imprime(matriz);
            }
        }
    }
}

```

```

    }
}

int main() {
    int n;
    cout << "Digite a ordem da matriz quadrada: ";
    cin >> n;

    vector<vector<double>> matriz(n, vector<double>(n));
    cout << "Digite os elementos da matriz:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matriz[i][j];
        }
    }

    GaussJordan(matriz);

    cout << "Matriz resultante:" << endl;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if(matriz[i][j] == -0){
                matriz[i][j] = 0;
            }
        }
    }

    imprime(matriz);

    return 0;
}

```

Compilador:


```
Digite a ordem da matriz quadrada: 3
Digite os elementos da matriz:
1
2
3
4
5
6
7
8
9
Matriz Inicial:
1 2 3
4 5 6
7 8 9

Pivoteamento Parcial (linha 1 trocada com linha 3):
7 8 9
4 5 6
1 2 3

Escalonamento (linha 1 dividida por 7):
1 1 1
4 5 6
1 2 3

Eliminacao (linha 2 subtraida de 4 vezes a linha 1):
1 1 1
0 0 1
1 2 3

Eliminacao (linha 3 subtraida de 1 vezes a linha 1):
1 1 1
0 0 1
0 1 2

Pivoteamento Parcial (linha 2 trocada com linha 3):
1 1 1
0 1 2
0 0 1
```

Escalonamento (linha 2 dividida por 1):

```
1 1 1
0 1 2
0 0 1
```

Eliminacao (linha 1 subtraida de 1 vezes a linha 2):

```
1 0 -1
0 1 2
0 0 1
```

Eliminacao (linha 3 subtraida de 0 vezes a linha 2):

```
1 0 -1
0 1 2
0 0 -0
```

Pivoteamento Parcial (linha 3 trocada com linha 3):

```
1 0 -1
0 1 2
0 0 -0
```

Escalonamento (linha 3 dividida por -0):

```
1 0 -1
0 1 2
-0 -0 1
```

Eliminacao (linha 1 subtraida de -1 vezes a linha 3):

```
1 0 0
0 1 2
-0 -0 1
```

Eliminacao (linha 2 subtraida de 2 vezes a linha 3):

```
1 0 0
0 1 0
-0 -0 1
```

Matriz resultante:

```
1 0 0
0 1 0
0 0 1
```