Matrix Calculator

Liapkin Rostyslav
Student prvního ročníku
MFF

Dokumentace pro uživatele

Dostupné programové funkce:

1. Inverze matice
2. Determinant matice
3. Násobení matic
4. Součet matic
5. Řešení krok za krokem
6. Použití výsledku výpočtů pro další výpočty

```
1 to set matrix A
2 to set matrix B
3 to inverze matrix A
4 to get determinant of matrix A
5 for matrix addition
6 for matrix multiplication(A*B)
7 to insert result in A
```

Používání programu:

1. Spusťte program "main.py"
2. Zadejte požadované matice

```
1 to set matrix A
2 to set matrix B
1
Enter count of rows
2
Enter count of columns
2
Enter row 1:
1 3
Enter row 2:
4 5

1 3
4 5

Matrix A created successfully
```

3. Použijte požadované akce v programu zadáním požadovaných čísel.
4. Program ukončíte zadáním "-1" v hlavním menu programu

```
1 to set matrix A
2 to set matrix B
3 to inverze matrix A
4 to get determinant of matrix A
-1 to end
-1

Process finished with exit code 0
```

Programátorská dokumentace

Hlavní soubor programu je "main.py", po každé akci se vykreslí menu, dokud uživatel nezadá "-1". Volá funkci "menu(isMatrixA, isMatrixB, isResult)", která v závislosti na tom, zda uživatel již nastavil matice a má-li nějaké výsledky, zobrazí menu.

```
Rostyslav *
def menu(isMatrixA, isMatrixB, isResult):
    print("1 to set matrix A")
    print("2 to set matrix B")
    if isMatrixA:
        print("3 to inverze matrix A")
        print("4 to get determinant of matrix A")
    if isMatrixA and isMatrixB:
        print("5 for matrix addition")
        print("6 for matrix multiplication(A*B)")
    if isResult != False:
        print("7 to insert result in A")
    print("-1 to end")
```

Výsledek zadaný uživatelem je uložen v proměnné "action". Pokud uživatel nezadá číslo nebo zadá špatné číslo, software ho o tom informuje.

```
try:
    action = int(input())
except:
    print("It is not a number")
    print()
```

1. action = 1. Volá funkce “setMatrix()”

```
match action:
    case 1:
        try:
            matrixA = CreateMatrix.setMatrix()
            print("Matrix A created successfully")
            print()
            isMatrixA = True
        except:
            print("You have made a mistake, try again")
            print()
```

Ve funkci uživatel vytvoří matici A.

```
def setMatrix():
    print("Enter count of rows")
    rows = int(input())
    print("Enter count of columns")
    columns = int(input())
    data = []
    for i in range(0, rows):
        print("Enter row " + str(i + 1) + ":")
        a = [int(x) for x in input().split()]
        data.append(a)
    matrix = Matrix(rows, columns, data)
    print()
    matrix.printMatrix()
    print()
    return matrix
```

Jestli uživatel udělal chybu, možná udělat matrici znovu.

Matice je vytvořena jako objekt třídy "Matrix", které lze podle potřeby upravit a má také funkci výstupu na konzoli

```
Rostyslav
class Matrix:
    Rostyslav
    def __init__(self, rows, columns, data):
        self.rows = rows
        self.columns = columns
        self.data = data

    Rostyslav
    def getRows(self):
        return self.rows

    Rostyslav
    def getColumns(self):
        return self.columns

    Rostyslav
    def getData(self):
        return self.data

    Rostyslav
    def printMatrix(self):
        for i in range(0, self.rows):
            row = ""
            for j in range(0, self.columns):
                row += str(self.data[i][j]) + " "
            print(row)
```

2. action = 2. Úplně stejně, ale pro matici B

```
case 2:
    try:
        matrixB = CreateMatrix.setMatrix()
        print("Matrix B created successfully")
        print()
        isMatrixB = True
    except:
        print("You have made a mistake, try again")
```

3. action = 3. Volá funkce “inversion(matrix)”, která potřebuje na vstup objekt třídy "Matrix"

```
case 3:
    if isMatrixA:
        isResult = MatrixInverse.inversion(matrixA)
        print()
    else:
        print("You need to set matrix A")
        print()
```

Toto je obvyklá Gaussova eliminace. Pokud inverzní matice neexistuje, uživatel obdrží odpověď, že není možné najít inverzní matici. V kódu jsou komentáře, abyste pochopili, jak funkce funguje. Funkce je příliš dlouhá na to, abych sem mohl vložit celý kód.

```
New Step
0 0 1 0
0 0 0 1

Impossible to solve
```

4. action = 4. Volá funkce “matrixDet(matrix)”, která potřebuje na vstup objekt třídy "Matrix".

```
case 4:
    if isMatrixA:
        DeterminantofaMatrix.matrixDet(matrixA)
    else:
        print("You need to set matrix A")
        print()
```

Pokud lze determinant nalézt, volá se funkce "determinant".

```
def matrixDet(matrix):
    data = matrix.getData()
    if matrix.getRows() == matrix.getColumns():
        print("Result is: " + str(determinant(data)))
    else:
        print("Matrix need to have the same count of columns and rows")
```

Je to rekurzivní funkci, která volá funkci minoritního rozkladu minor(data)

```
def determinant(data):  # Recursive function
    size = len(data)
    if size == 2:   # if size is 2, we can just count determinant
        return det2(data)

    return sum((-1) ** j * data[0][j] * determinant(minor(data, 0, j))
               for j in range(size))
```

Zde je vlastní funkce

```
Rostyslav
def minor(data, i, j):  # getting minor of matrix
    tmp = [row for k, row in enumerate(data) if k != i]
    tmp = [col for k, col in enumerate(zip(*tmp)) if k != j]
    return tmp
```

5. action = 5. Volá funkce “addMatrices(matrixA, matrixB)”, která potřebuje na vstup dva objekty třídy "Matrix".

```
case 5:
    if isMatrixA and isMatrixB:
        isResult = MatrixAddition.addMatrices(matrixA, matrixB)
        print()
    else:
        print("You need to create both matrices")
```

Tato funkce zkontroluje, zda lze matice sčítat, a sečte je.

```
def addMatrices(matrixA, matrixB):
    if matrixA.getRows() == matrixB.getRows() and matrixA.getColumns() == matrixB.getColumns():
        dataA = matrixA.getData()
        dataB = matrixB.getData()
        dataRes = []
        print("RESULT")
        for i in range(0, matrixA.getRows()):
            rowRes = ""
            row = []
            for j in range(0, matrixA.getColumns()):
                row.append(dataA[i][j] + dataB[i][j])
                rowRes += str(dataA[i][j] + dataB[i][j]) + " "
            print(rowRes)
            dataRes.append(row)

        return Matrix(matrixA.getRows(), matrixB.getColumns(), dataRes)
    else:
        print("Count of columns and rows of both matrices must be the same")
        return False
```

6. action = 6. Volá funkce “mulitply(matrixA, matrixB)”, která potřebuje na vstup dva objekty třídy "Matrix". Tato funkce zkontroluje, zda lze násobit matice, a vynásobí je.

```
def multiply(matrixA, matrixB):
    if matrixA.getColumns() == matrixB.getRows():
        dataA = matrixA.getData()
        dataB = matrixB.getData()
        dataResult = []
        stepByStep = []
        print("RESULT")
        for i in range(0, matrixA.getRows()):
            rowRes = ""
            row = []

            for j in range(0, matrixB.getColumns()):
                buffer = 0
                oneStep = "a" + str(i + 1) + str(j + 1)+"="
                for k in range(0, matrixA.getColumns()):
                    buffer = buffer + dataA[i][k] * dataB[k][j]
                    if k == 0:
                        oneStep += str(dataA[i][k]) + "*" + str(dataB[k][j])
                    else:
                        oneStep +="+" + str(dataA[i][k]) + "*" + str(dataB[k][j])
                row.append(buffer)
                rowRes += str(buffer) + " "
                stepByStep.append(oneStep)
            dataResult.append(row)
            print(rowRes)
        print()
        print("STEP-BY-STEP")
        for i in range_(0, len(stepByStep)):
            print(stepByStep[i])
        return Matrix(matrixA.getRows(), matrixB.getColumns(), dataResult)
    else:
        print("Count of columns of matrix A must be the same as count of rows matrix B")
        return False
```

7. action = 7, zapíše výsledek předchozích výpočtů do matice A

```
case 7:
    if isResult != False:
        matrixA = isResult
        print("Successfully inserted result into A")
        matrixA.printMatrix()
        print()
    else:
        print("You need to get result")
```

8. action = -1. Právě končí program

```
case -1:
    break
```

Jak funguje řešení krok za krokem? Jednoduše zobrazuje výsledky výpočtů tak, jak by je zobrazil člověk.