# Введение

Генератор заданий по матричным операциям предоставляет библиотеку (набор библиотек) для работы с матрицами на языке C#. Большое внимание уделено функциям, позволяющим создавать учебные задания по работе с матрицами, особенно задание на вычисление матрицы по формуле. Также разработаны функции, упрощающие представление объектов, результатов вычислений и прочих операций над матрицами в формате LaTeX.

# Задания

## Матрицы

### Matrix

Предназначен для работы с матрицами.

Поля

|  |  |  |
| --- | --- | --- |
| Field | Matrix<double> | непосредственно матрица с коэффициентами в формате числа с плавающей точкой |
| Label | string | название матрицы |

Методы

|  |  |
| --- | --- |
| DeterminantFind() | возвращает значение детерминанта матрицы |
| GetColumnCout() | возвращает количество столбцов матрицы |
| GetRowCount() | возвращает количество строк матриц |
| InverseMatrix() | возвращает обратную матрицу |
| MakeRealZeroDet() | преобразует матрицу с действительными коэффициентами в матрицу с определителем, равным нулю (усиленная версия функции MakeZeroDeterminant) |
| MakeZeroDeterminant() | преобразует матрицу с целочисленными коэффициентами в матрицу с определителем, равным нулю |
| Matrix() | конструктор по умолчанию, создающий пустую матрицу без названия |
| Matrix(int XSize, int YSize, double MinValue, double MaxValue, bool real, bool zeroDet,string name) | конструктор, создающий матрицу со следующими параметрами:   * количество строк и столбцов; * максимальное и минимальное значение элементов; * наличие вещественных коэффициентов; * определитель равный/не равный нулю; * название. |
| Matrix(Matrix<double> Field, string Name) | конструктор, создающий матрицу из матрицы и названия |
| OutToString() | возвращает текстовое представление матрицы (например, для вывода в консоль) |
| SetMatrix(double[,] DoubleMatrix) | заполняем матрицу из массива чисел |
| ToTEX() | преобразует матрицу в формат LaTeX для записи в файл \*.lth |
| TranspositionMatrix() | возвращает транспонированную матрицу |

### NameComparer

Класс, унаследованный от IComparer, предназначен для сравнения матриц по названию.

Методы

|  |  |
| --- | --- |
| Compare([Matrix](#_Matrix_1) X, [Matrix](#_Matrix_1) Y) | сравнивает две матрицы по названию |

### TreeToTex

Преобразует дерево матриц в матричное выражение в формате LaTeX

Методы

|  |  |
| --- | --- |
| PrintTree([Tree](#_Tree) \_tree) | возвращает строку с матричным выражением в формате LaTeX |
| Print([Node](#_Node_1) N, ref string rez) | возвращает строку с матричным выражением в формате LaTeX из поддерева с корнем в данном узле |

## Задание

### Template

Содержит параметризованное описание задания

Поля

|  |  |  |
| --- | --- | --- |
| Id\_type | int | тип задания |
| title | string | название задания |
| typeTitle | string | текстовая запись типа задания |
| number | int | номер задания |
| comment | string[] | текстовый комментарий к заданию |
| matrixCount | int | количество матриц |
| elementType | [ElementType](#_ElementType) | тип элементов матриц |
| matrixSizeType | [MatrixSizeType](#_MatrixSizeType) | описывает, каким образом определяются матрицы |
| countType | [CountType](#_CountType) | резервное поле (не используется) |
| firstMatrix | bool | наличие первой матрицы с пользовательской размерностью |
| firstMatrixM | int? | количество строк в первой матрице |
| firstMatrixN | int? | количество столбцов в первой матрице |
| secondMatrix | bool | наличие второй матрицы с пользовательской размерностью |
| secondMatrixM | int? | количество строк во второй матрице |
| secondMatrixN | int? | количество столбцов во второй матрице |
| thirdMatrix | bool | наличие третьей матрицы с пользовательской размерностью |
| thirdMatrixM | int? | количество строк в третьей матрице |
| thirdMatrixN | int? | количество столбцов в третьей матрице |
| vectorExist | bool | наличие вектор-строки/столбца |
| quadExist | bool | наличие квадратной матрицы |
| byFormulaFind | bool | наличие задания вычисления формулы |
| formulaOutM | int? | количество строк в итоговой матрице при вычислении формулы |
| formulaOutN | int? | количество столбцов в итоговой матрице при вычислении формулы |
| numberExist | bool | наличие числа в формуле |
| determMatrix | bool | наличие задания нахождения определителя матрицы |
| determMatrixType | [DetermenantType](#_DetermenantType) | тип задания нахождения определителя матрицы |
| invertMatrix | bool | наличие задания нахождения обратной матрицы |
| invertMatrixType | [InvertType](#_InvertType) | тип задания нахождения обратной матрицы |
| transeMatrix | bool | наличие задания транспонирования матрицы |
| transeMatrixType | [TransposeType](#_TransposeType) | тип задания транспонирования матрицы |
| countRow | bool | тип задания нахождения количества строк матрицы |
| countRowType | [CountType](#_CountType) | наличие задания нахождения количества строк матрицы |
| countColumn | bool | тип задания нахождения количества столбцов матрицы |
| countColumnType | [CountType](#_CountType) | наличие задания нахождения количества столбцов матрицы |
| outToScreen | bool | наличие задания вывода на экран элемента матрицы |
| outRowByNumber | bool | наличие задания вывода на экран строки матрицы |
| outColumnByNumber | bool | наличие задания вывода на экран столбца матрицы |
| outSubMatrix | bool | наличие задания вывода на экран подматрицы |
| outSubMatrixType | [SubMatrixType](#_SubMatrixType) | тип задания вывода на экран подматрицы |
| calcValue | bool | наличие задания вычисления значения |
| calcValueOperation | [CalculationType](#_CalculationType) | операция в задании вычисления значения |
| calcValueType | [ElementValueType](#_ElementValueType) | тип элементов в задании вычисления значения |
| calcValueArea | [SubMatrixType](#_SubMatrixType) | тип подматрицы в задании вычисления значения |
| differSizeSum | bool | наличие задания на нахождение суммы матриц с несовпадающими размерностями |
| differSizeMult | bool | наличие задания на нахождение произведения матриц с несоответствующими размерностями |

Методы

|  |  |
| --- | --- |
| Serialize(string XMLFilePath) | сохранение задания в XML-файл |
| DeSerialize(string XMLFilePath) | загрузка задания из XML-файла |

### ElementType

Перечисление типов элементов

|  |  |
| --- | --- |
| positive | положительные |
| any | и положительные, и отрицательные |

### MatrixSizeType

Перечисление способов определения матриц

|  |  |
| --- | --- |
| random | случайным образом |
| formula | по формуле |
| custom | пользовательские параметры |

### DetermenantType

Перечисление типов задания на нахождение определителя матрицы

|  |  |
| --- | --- |
| allMatrix | для всех матриц |
| singleMatrix | для одной матрицы |
| quadMatrix | для квадратной матрицы |

### InvertType

Перечисление типов задания на нахождение обратной матрицы

|  |  |
| --- | --- |
| allMatrix | для всех матриц |
| singleMatrix | для одной матрицы |
| quadMatrix | для квадратной матрицы |
| zeroDetMatrix | для матрицы с нулевым определителем |

### TransposeType

Перечисление типов задания на транспонирование матрицы

|  |  |
| --- | --- |
| allMatrix | для всех матриц |
| singleMatrix | для одной матрицы |

### CountType

Перечисление типов задания на нахождение количества строк или столбцов матрицы

|  |  |
| --- | --- |
| allMatrix | для всех матриц |
| singleMatrix | для одной матрицы |

### SubMatrixType

Перечисление типов подматриц

|  |  |
| --- | --- |
| MainDiagonal | главная диагональ |
| SecondDiagonal | побочная диагональ |
| FewRow | несколько строк |
| FewColumn | несколько столбцов |
| SubMatrix | подматрица (пересечение строк и столбцов) |
| SingleRow | одна строка |
| SingleColumn | один столбец |
| AllMatrix | вся матрица |

### CalculationType

Перечисление типов операции в задании вычисления значения

|  |  |
| --- | --- |
| Max | наибольшее значение среди элементов |
| Min | наименьшее значение среди элементов |
| Sum | сумма значений элементов |
| Avg | среднее арифметическое значений элементов |
| Count | количество элементов |

### ElementValueType

Перечисление типов элементов в задании вычисления значения

|  |  |
| --- | --- |
| Positive | положительные элементы |
| Negative | отрицательные элементы |
| Odd | нечётные элементы |
| Even | чётные элементы |
| Number | элементы, равные числу (нулю) |
| All | все элементы |

## Генератор заданий и решений

### TaskGenerator

Создаёт задание и решение на нахождение матрицы.

Поля

|  |  |  |
| --- | --- | --- |
| template | [Template](#_Template) | шаблон задания |
| templatePath | string | путь к файлу шаблона задания |
| maketPath | string | путь к файлу индивидуального задания |
| solvePath | string | путь к файлу решения индивидуального задания |
| creator | MatrixCreator | экземпляр модуля составления заданий и решений |
| matrixList | List<[Matrix](#_Matrix)> | список матриц |

### MatrixCreator

Создаёт задание и решение на нахождение матрицы

Поля

|  |  |  |
| --- | --- | --- |
| random | MersenneTwister | генератор случайных чисел |
| matrixList | List<[Matrix](#_Matrix)> | список матриц |
| template | [Template](#_Template_3) | шаблон задания |
| taskText | List<string> | текст задания |
| solveText | List<string> | текст решения |
| taskCount | int | количество заданий |

Методы

|  |  |
| --- | --- |
| Generate | общий метод генерации заданий |
| GenerateWithFormula | генерация матриц по формуле |
| GenerateCustomMatrix(List<int[]> matrixSize) | генерация матриц пользовательским параметрам |
| GenerateRandom | генерация матриц случайным образом |
| SetMatrixName | именование матриц |
| AddMatrixTask | добавление заданий |
| GetRandomMatix(List<[Matrix](#_Matrix)> matrixList) | получение случайной матрицы |
| GetTask | получение текста задания |
| GetSolve | получение текста решения |

### RandDecide

Генератор случайного решения

Методы

|  |  |
| --- | --- |
| Decide(bool isQuad, int mult, int sum, int transe, int inverse) | возвращает случайное решение при заданных коэффициентах матричных операций |

Используется метод расширения IsInclude для Tuple<int, int>, позволяющий определять принадлежность числа диапазону

### OperationEnum

Перечисление операций и их типов

|  |  |
| --- | --- |
| mult | операция умножения |
| sum | операция сложения |
| transe | операция транспонирования матрицы |
| inverse | операция нахождения обратной матрицы |
| error | ошибочная операция |
| empty | пустая операция |
| unitary | унарная операция |
| bynary | бинарная операция |
| any | произвольная операция (любая) |

### MatrixParam

Содержит параметры задания на нахождение матрицы по формуле. Использовать не рекомендуется, потому что является устаревшим.

Поля

|  |  |  |
| --- | --- | --- |
| MatrixCount | int | количество матриц в формуле |
| OutMatrixRank | int | разрядность итоговой матрицы |
| TaskCount | int | количество дополнительных заданий |
| NumberExist | bool | наличие числа в формуле |
| RealNumber | bool | наличие вещественных чисел |
| Vectors | bool | наличие вектор-строки и вектор-столбца |
| QuadMatrix | bool | наличие квадратной матрицы |
| DifficultTask | bool | наличие заданий повышенной сложности (не используется) |
| DifficultTasks | bool[] | наличие заданий повышенной сложности (см. DifficultyTaskType) |
| Description | List<string> | комментарий к заданию |

Методы

|  |  |
| --- | --- |
| MatrixParam(int matrixCount, int outMatrixRank, int taskCount, bool numberExist, bool realNumber, bool quadMatrix, bool vectors, bool difficultTask, bool[] difficultTasks, List<string> description) | конструктор, создающий задание по параметрам, указанным в атрибутах класса |

### DifficultyTaskType

Перечисление заданий дополнительной сложности

|  |  |  |
| --- | --- | --- |
| InvertZeroDiscriminant | 0 | нахождение обратной матрицы для матрицы с нулевым определителем |
| InvalidSum | 1 | нахождение суммы матриц с разными размерностями |
| InvalidMult | 2 | нахождение произведения матриц с несоответствующими размерностями |

### Дерево формулы

#### Node

Абстрактный класс вершины дерева вычисления формулы

Поля

|  |  |  |
| --- | --- | --- |
| parent | [OperationNode](#_OperationNode) | родительская вершина-операция |

#### OperationNode

Абстрактный класс вершины для произвольной операции

Поля

|  |  |  |
| --- | --- | --- |
| isBracket | bool | наличие скобок |
| operation | [OperationEnum](#_OperationEnum) | матричная операция |

#### UnaryOperation

Абстрактный класс вершины унарной операции, производный от класса [OperationNode](#_OperationNode)

Поля

|  |  |  |
| --- | --- | --- |
| child | [Node](#_Node) | операнд (вершина) |

Методы

|  |  |
| --- | --- |
| UnaryOperation | конструктор по умолчанию |

#### InvertOperation

Класс вершины операции нахождения обратной матрицы, производный от класса [UnaryOperation](#_UnaryOperation)

Методы

|  |  |
| --- | --- |
| InvertOperation | конструктор по умолчанию |

#### TranseOperation

Класс вершины операции транспонирования, производный от класса [UnaryOperation](#_UnaryOperation)

Методы

|  |  |
| --- | --- |
| TranseOperation | конструктор по умолчанию |

#### BynaryOperation

Абстрактный класс вершины бинарной операции, производный от класса [OperationNode](#_OperationNode)

Поля

|  |  |  |
| --- | --- | --- |
| firstChild | [Node](#_Node) | первый операнд (вершина) |
| secondChild | [Node](#_Node) | второй операнд (вершина) |

Методы

|  |  |
| --- | --- |
| BynaryOperation | конструктор по умолчанию |

#### MultOperation

Класс вершины операции умножения, производный от класса [UnaryOperation](#_UnaryOperation)

Методы

|  |  |
| --- | --- |
| MultOperation | конструктор по умолчанию |

#### SumOperation

Класс вершины операции сложения, производный от класса [UnaryOperation](#_UnaryOperation)

Методы

|  |  |
| --- | --- |
| SumOperation | конструктор по умолчанию |

#### BasicNode

Абстрактный класс вершины (матрицы или операции), производный от класса [Node](#_Node). Операции соответствует матрица, которая получается в результате её вычисления.

Поля

|  |  |  |
| --- | --- | --- |
| label | char | название матрицы |
| nSize | int | количество строк матрицы |
| mSize | int | количество столбцов матрицы |
| matrix | [Matrix](#_Matrix) | матрица |

Методы

|  |  |
| --- | --- |
| isQuad | определяет, является ли матрица вершины квадратной |

#### Number

Класс вершины-числа, производный от класса [Node](#_Node)

Поля

|  |  |  |
| --- | --- | --- |
| numb | double | значение числа |

#### BasicMatrixNode

Абстрактный класс вершины-матрицы, производный от класса [BasicNode](#_BasicNode)

Поля

|  |  |  |
| --- | --- | --- |
| isClosed | bool | определяет, можно ли преобразовывать данную вершину или нет (резервное поле, в проекте не используется) |

#### Vector

Абстрактный класс вершины-вектор-строки/столбца, производный от класса [BasicNode](#_BasicNode)

#### Row

Класс вершины-вектор-строки, производный от класса [Vector](#_Vector)

#### Column

Класс вершины-вектор-столбца, производный от класса [Vector](#_Vector)

#### QuadMatrix

Класс вершины-квадратной матрицы, производный от класса [BasicNode](#_BasicNode)

#### GenericMatrix

Класс вершины-матрицы, производный от класса [BasicNode](#_BasicNode)

#### Tree

Дерево вычисления формулы

Поля

|  |  |  |
| --- | --- | --- |
| root | [BasicNode](#_CalcNode) | корневая вершина дерева |
| isQuad | bool | определяет, является ли корневая матрица квадратной |
| nodes | List<[Node](#_Node_1)> | список вершин дерева |
| random | MersenneTwister | генератор случайных чисел |
| numberExist | bool | задаёт наличие числа в формуле |
| quadExist | bool | задаёт наличие в формуле квадратной матрицы |
| vectorExist | bool | задаёт наличие в формуле вектор-строки/столбца |
| zeroDet | bool | задаёт наличие в формуле квадратной матрицы с нулевым определителем |
| qMax | [QuadMatrix](#_QuadMatrix) | вершина-квадратная матрица |
| numb | [Number](#_Number) | вершина-число |
| \_mSize | int | количество строк результирующей матрицы |
| \_nSize | int | количество столбцов результирующей матрицы |

Методы

|  |  |
| --- | --- |
| Tree() | конструктор по умолчанию |
| GetFormulaTeX() | возвращает текстовое представление формулы в формате TeX |
| GetOutNatrix() | возвращает текстовое представление результирующей матрицы в формате TeX |
| Build(int matrixCount, [ElementType](#_ElementType) elementType) | строит дерево вычисления формулы с заданным количеством матриц и типом элементов |
| SetSize([Node](#_Node_1) nod, int m, int n) | устанавливает размерности дочерних вершин выбранной вершины с указанным количеством строк и столбцов |
| Tree ([GenericMatrix](#_GenericMatrix) matrix) | инициализирует дерево из указанной матрицы |
| AddNumber() | добавляет к дереву вершину-число |
| AddVectors() | добавляет к дереву вершины-вектор-строку и -число |
| AddQuad() | добавляет к дереву вершину-квадратную матрицу |
| AddToRoot() | случайным образом добавляет корневую унарную операцию |
| NodeOpen([BasicNode](#_CalcNode) node) | преобразует вершину в матричную операцию |
| AllType<T>() | возвращает все вершины указанного типа |
| SetName() | именует матрицы и устанавливает значение числа, если оно присутствует в формуле |
| FillMatrix([ElementType](#_ElementType) elementType) | устанавливает значения элементов матриц |
| Calc([OperationNode](#_OperationNode) node) | вычисляет матрицу данной вершины дерева и её потомков |
| GetMaxSize([BasicNode](#_CalcNode) N) | возвращает количество столбцов первой матрицы и количество строк второй матрицы для вершины-операции умножения |
| ValueConstraint(double value) | проверяет элемент матрицы на принадлежность диапазону допустимых значений |

## Задания

### IMatrixOperation

Интерфейс, описывающий методы, общие для матричных операций

Методы

|  |  |
| --- | --- |
| Operate() | выполняет матричную операцию и возвращает её результат |
| SetMatrix([Matrix](#_Matrix_1) Matrix) | задаёт матрицу для выполнения операции |
| ToText() | возвращает текст, соответствующий заданию на выполнение данной операции |

### BasicOperation

Абстрактный класс операций над матрицами, входящих в задание

Поля

|  |  |  |
| --- | --- | --- |
| Matrix | [Matrix](#_Matrix_1) | Матрица, над которой выполняются операции |

Методы

|  |  |
| --- | --- |
| SetMatrix([Matrix](#_Matrix_1) Matrix) | задаёт матрицу для выполнения операции |
| BasicOperation() | конструктор по умолчанию |
| BasicOperation([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для выполнения операции |

### FindDeter

Класс операции нахождения определителя матрицы, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Методы

|  |  |
| --- | --- |
| FindDeter() | конструктор по умолчанию |
| FindDeter([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для нахождения определителя |
| Operate() | вычисляет определитель матрицы и возвращает его значение |
| ToText() | возвращает текст, соответствующий заданию на нахождение определителя матрицы |

### Inverse

Класс операции нахождения обратной матрицы, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Методы

|  |  |
| --- | --- |
| Inverse () | конструктор по умолчанию |
| Inverse ([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для нахождения обратной матрицы |
| Operate() | вычисляет и возвращает матрицу, обратную данной |
| ToText() | возвращает текст, соответствующий заданию на нахождение обратной матрицы |

### Transposition

Класс операции транспонирования матрицы, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Методы

|  |  |
| --- | --- |
| Transposition () | конструктор по умолчанию |
| Transposition ([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для транспонирования |
| Operate() | вычисляет и возвращает транспонированную матрицу |
| ToText() | возвращает текст, соответствующий заданию на транспонирование матрицы |

### GetRowCount

Класс операции нахождения количества строк матрицы, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Методы

|  |  |
| --- | --- |
| GetRowCount () | конструктор по умолчанию |
| GetRowCount ([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для нахождения количества строк |
| Operate() | возвращает количество строк матрицы |
| ToText() | возвращает текст, соответствующий заданию на нахождение количества строк матрицы |

### GetColumnCout

Класс операции нахождения количества столбцов матрицы, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Методы

|  |  |
| --- | --- |
| GetColumnCout () | конструктор по умолчанию |
| GetColumnCout ([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для нахождения количества столбцов |
| Operate() | возвращает количество столбцов матрицы |
| ToText() | возвращает текст, соответствующий заданию на нахождение количества столбцов матрицы |

### GetElement

Класс операции выбора фрагмента матрицы, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Атрибуты

|  |  |  |
| --- | --- | --- |
| Selector | [IAreaSelector](#_IAreaSelector) | селектор – определяет фрагмент, извлекаемый из матрицы |
| OutText | string | строка с текстом задания |

Методы

|  |  |
| --- | --- |
| GetElement () | конструктор по умолчанию |
| GetElement([IAreaSelector](#_IAreaSelector) Selector, Matrix Matrix) | конструктор, задающий селектор и матрицу для выбора фрагмента |
| GetElement([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для выбора фрагмента и создающий селектор случайным образом |
| Operate() | возвращает количество столбцов матрицы |
| ToText() | возвращает текст, соответствующий заданию на нахождение количества столбцов матрицы |

### MatrixCalculator

Класс операции вычисления значения, производный от [BasicOperation](#_BasicOperation_1). Реализует интерфейс [IMatrixOperation](#_IMatrixOperation_1).

Атрибуты

|  |  |  |
| --- | --- | --- |
| AreaSelector | [IAreaSelector](#_IAreaSelector) | селектор, определяющий фрагмент, извлекаемый из матрицы |
| SetSelector | [ISetSelector](#_ISetSelector_1) | селектор, определяющий тип извлекаемых элементов |
| Operation | [ISelectOperation](#_ISelectOperation_1) | операция, применяемая к элементам |

Методы

|  |  |
| --- | --- |
| MatrixCalculator() | конструктор по умолчанию, задающий селекторы и операцию случайным образом |
| MatrixCalculator([IAreaSelector](#_IAreaSelector) AreaSelector, [ISetSelector](#_ISetSelector_1) SetSelector, [ISelectOperation](#_ISelectOperation_1) Operation) | конструктор, задающий селекторы и операцию для вычисления значения |
| MatrixCalculator([Matrix](#_Matrix_1) Matrix) | конструктор, задающий матрицу для вычисления значения |
| Operate() | возвращает вычисленное значение |
| ToText() | возвращает текст, соответствующий заданию на вычисление значения |

### Задания повышенной сложности

#### IBynaryOperation

Интерфейс, описывающий методы, общие для бинарных операций.

Поля

|  |  |  |
| --- | --- | --- |
| solveText | string | текст решения |
| taskText | string | текст задания |
| firstMatrix | [Matrix](#_Matrix_1) | первый операнд |
| secondMatrix | [Matrix](#_Matrix_1) | второй операнд |
| result | [Matrix](#_Matrix_1) | результат операции |

#### BynaryOperation

Абстрактный класс бинарной операции, реализует интерфейс [IBynaryOperation](#_IBynaryOperation)

Поля

|  |  |  |
| --- | --- | --- |
| \_matrixList | List<[Matrix](#_Matrix_1)> | список матриц |
| solveText | string | текст решения |
| taskText | string | текст задания |
| firstMatrix | [Matrix](#_Matrix_1) | первый операнд |
| secondMatrix | [Matrix](#_Matrix_1) | второй операнд |
| result | [Matrix](#_Matrix_1) | результат операции |
| isComplete | bool | определяет, выполнена ли операция |

Методы

|  |  |
| --- | --- |
| Operate() | абстрактный метод выполнения операции |
| BasicBynOperation(List<[Matrix](#_Matrix_1)> matrixList) | конструктор |

#### InvalidMult

Класс операции умножения, производный от [BynaryOperation](#_BynaryOperation)

Методы

|  |  |
| --- | --- |
| Operate() | выполнения операции |
| InvalidMult(List<[Matrix](#_Matrix_1)> matrixList) | конструктор |

#### InvalidSum

Класс операции сложения, производный от [BynaryOperation](#_BynaryOperation)

Методы

|  |  |
| --- | --- |
| Operate() | выполнения операции |
| InvalidSum (List<[Matrix](#_Matrix_1)> matrixList) | конструктор |

## Операции нахождения значения

### ISelectOperation

Интерфейс, описывающий методы, общие для операций, применимых к подматрицам

Методы

|  |  |
| --- | --- |
| Operate(List<double> Matrix) | возвращает результат выполнения операции над списком элементов |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции |

### SumOfElement

Класс операции вычисления суммы элементов. Реализует интерфейс [ISelectOperation](#_ISelectOperation_1).

Методы

|  |  |
| --- | --- |
| Operate(List<double> Matrix) | возвращает сумму элементов списка |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции вычисления суммы элементов |

### MinElement

Класс операции вычисления наименьшего элемента. Реализует интерфейс [ISelectOperation](#_ISelectOperation_1)

Методы

|  |  |
| --- | --- |
| Operate(List<double> Matrix) | возвращает наименьший элемент из списка |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции нахождения наименьшего элемента |

### MaxElement

Класс операции вычисления наибольшего элемента. Реализует интерфейс [ISelectOperation](#_ISelectOperation_1)

Методы

|  |  |
| --- | --- |
| Operate(List<double> Matrix) | возвращает наибольший элемент из списка |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции нахождения наибольшего элемента |

### AvgOfElement

Класс операции вычисления среднего арифметического элементов. Реализует интерфейс [ISelectOperation](#_ISelectOperation_1)

Методы

|  |  |
| --- | --- |
| Operate(List<double> Matrix) | возвращает среднее арифметическое списка элементов |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции вычисления среднего арифметического |

### CountOfElement

Класс операции определения количества элементов. Реализует интерфейс [ISelectOperation](#_ISelectOperation_1).

Методы

|  |  |
| --- | --- |
| Operate(List<double> Matrix) | возвращает количество элементов списка |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции определения количества элементов |

## Типы элементов

### ISetSelector

Интерфейс, описывающий методы, извлекающие подмножество элементов в зависимости от их арифметических свойств

Методы

|  |  |
| --- | --- |
| GetSet(Matrix<double> Matrix) | возвращает подмножество элементов из матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции |

### PositiveElements

Класс извлечения положительных элементов из матрицы. Реализует интерфейс [ISetSelector](#_ISetSelector_1).

Методы

|  |  |
| --- | --- |
| GetSet(Matrix<double> Matrix) | возвращает список положительных элементов матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению положительных значений |

### NegativeElements

Класс извлечения отрицательных элементов из матрицы. Реализует интерфейс [ISetSelector](#_ISetSelector_1).

Методы

|  |  |
| --- | --- |
| GetSet(Matrix<double> Matrix) | возвращает список отрицательных элементов матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению отрицательных значений |

### OddElements

Класс извлечения чётных элементов из матрицы. Реализует интерфейс [ISetSelector](#_ISetSelector_1).

Методы

|  |  |
| --- | --- |
| GetSet(Matrix<double> Matrix) | возвращает список чётных элементов матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению чётных значений |

### EvenElements

Класс извлечения нечётных элементов из матрицы. Реализует интерфейс [ISetSelector](#_ISetSelector_1).

Методы

|  |  |
| --- | --- |
| GetSet(Matrix<double> Matrix) | возвращает список нечётных элементов матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению нечётных значений |

### AllElements

Класс извлечения всех элементов из матрицы. Реализует интерфейс [ISetSelector](#_ISetSelector_1).

Методы

|  |  |
| --- | --- |
| GetSet(Matrix<double> Matrix) | возвращает список всех элементов матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению всех значений |

### NumberEqualElements

Класс извлечения из матрицы элементов, равных числу. Реализует интерфейс [ISetSelector](#_ISetSelector_1).

Поля

|  |  |  |
| --- | --- | --- |
| Number | double | число, с которым сравниваются элементы матрицы |

Методы

|  |  |
| --- | --- |
| NumberEqualElements(double Number) | задаёт число, с которым сравниваются элементы матрицы (по умолчанию – ноль) |
| GetSet(Matrix<double> Matrix) | возвращает список элементов, равных заданному числу |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов, равных числу |

## Типы селекторов

### IAreaSelector

Интерфейс, описывающий методы, извлекающие подмножество элементов в зависимости от их расположения

Методы

|  |  |
| --- | --- |
| GetArea(Matrix<double> Matrix) | возвращает подмножество элементов из матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего операции |

### MainDiagonalSelector

Класс извлечения элементов главной диагонали матрицы. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Методы

|  |  |
| --- | --- |
| GetArea(Matrix<double> Matrix) | возвращает список элементов главной диагонали матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов главной диагонали матрицы |

### SecondDiagonalSelector

Класс извлечения элементов побочной диагонали матрицы. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Методы

|  |  |
| --- | --- |
| GetArea(Matrix<double> Matrix) | возвращает список элементов побочной диагонали матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов побочной диагонали матрицы |

### SubMatrixSelector

Класс извлечения элементов на пересечении нескольких строк и нескольких столбцов. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Поля

|  |  |  |
| --- | --- | --- |
| IsCreated | bool | определяет, выполнена ли операция |
| StartRow | int | номер начальной строки |
| StartColumn | int | номер начального столбца |
| RowCount | int | количество строк |
| ColumnCount | int | количество столбцов |
| AllColumn | bool | указывает, задействованы ли все столбцы |
| AllRow | bool | указывает, задействованы ли все строки |

Методы

|  |  |
| --- | --- |
| SubMatrixSelector(int StartRow, int StartColumn, int RowCount, int ColumnCount,bool AllRow, bool AllColumn) | конструктор, использующий номера начальных строк и столбцов, количество необходимых строк и столбцов, флаговые переменные использования всех строк и всех столбцов |
| SubMatrixSelector() | конструктор по умолчанию |
| GetArea(Matrix<double> Matrix) | возвращает список элементов на пересечении нескольких строк и нескольких столбцов |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов на пересечении нескольких строк и нескольких столбцов |

### FewRowSelector

Класс извлечения элементов нескольких строк. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Поля

|  |  |  |
| --- | --- | --- |
| IsCreated | bool | определяет, выполнена ли операция |
| StartRow | int | номер начальной строки |
| RowCount | int | количество строк |
| AllRow | bool | указывает, задействованы ли все строки |
| SubMatrix | [IAreaSelector](#_ISetSelector_1) | селектор подматрицы |

Методы

|  |  |
| --- | --- |
| FewRowSelector(int StartRow, int RowCount,bool AllRow) | конструктор, использующий номер начальной строки, количество необходимых строк и флаговую переменную использования всех строк |
| FewRowSelector() | конструктор по умолчанию |
| GetArea(Matrix<double> Matrix) | возвращает список элементов нескольких строк |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов нескольких строк |

### SingleRowSelector

Класс извлечения элементов строки. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Поля

|  |  |  |
| --- | --- | --- |
| IsCreated | bool | определяет, выполнена ли операция |
| Row | int | номер строки |
| AllRow | bool | указывает, задействованы ли все строки |
| SubMatrix | [IAreaSelector](#_ISetSelector_1) | селектор подматрицы |

Методы

|  |  |
| --- | --- |
| SingleRowSelector(int Row,bool AllRow) | конструктор, использующий номер строки и флаговую переменную использования всех строк |
| SingleRowSelector() | конструктор по умолчанию |
| GetArea(Matrix<double> Matrix) | возвращает список элементов нескольких строк |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов нескольких строк |

### FewColumnSelector

Класс извлечения элементов нескольких столбцов. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Поля

|  |  |  |
| --- | --- | --- |
| IsCreated | bool | определяет, выполнена ли операция |
| StartColumn | int | номер начального столбца |
| ColumnCount | int | количество столбцов |
| AllColumn | bool | указывает, задействованы ли все столбцы |
| SubMatrix | [IAreaSelector](#_ISetSelector_1) | селектор подматрицы |

Методы

|  |  |
| --- | --- |
| FewColumnSelector(int StartColumn, int ColumnCount, bool AllColumn) | конструктор, использующий номер начального столбца, количество необходимых столбцов и флаговую переменную использования всех столбцов |
| FewColumnSelector() | конструктор по умолчанию |
| GetArea(Matrix<double> Matrix) | возвращает список элементов нескольких столбцов |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов нескольких столбцов |

### SingleColumnSelector

Класс извлечения элементов столбца. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Поля

|  |  |  |
| --- | --- | --- |
| IsCreated | bool | определяет, выполнена ли операция |
| Row | int | номер столбца |
| AllRow | bool | указывает, задействованы ли все столбцы |
| SubMatrix | [IAreaSelector](#_ISetSelector_1) | селектор подматрицы |

Методы

|  |  |
| --- | --- |
| SingleColumnSelector(int Column,bool AllColumn) | конструктор, использующий номер столбца и флаговую переменную использования всех столбцов |
| SingleColumnSelector() | конструктор по умолчанию |
| GetArea(Matrix<double> Matrix) | возвращает список элементов нескольких столбцов |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению элементов нескольких столбцов |

### AllMatrixSelector

Класс извлечения всех элементов матрицы. Реализует интерфейс [IAreaSelector](#_IAreaSelector).

Методы

|  |  |
| --- | --- |
| GetArea(Matrix<double> Matrix) | возвращает список всех элементов матрицы |
| ToText() | возвращает текстовое представление фрагмента задания, соответствующего нахождению всех элементов матрицы |

# Обработка ошибок

## ExcepthionHelper

Методы

|  |  |
| --- | --- |
| GetMessage(Exception ex) | возвращает сообщение об ошибке для пользователя |
| InvalidOperationException(Exception ex) | обработчик ошибок типа InvalidOperationException |
| UriFormatException(Exception ex) | обработчик ошибок типа UriFormatException |
| ArgumentException(Exception ex) | обработчик ошибок типа ArgumentException |
| MySQLException(Exception ex) | обработчик ошибок типа MySqlException |
| WebException(Exception ex) | обработчик ошибок типа WebException |
| Exception(Exception ex) | обработчик прочих ошибок типа Exception |

# Модуль работы с сервисом учёта контингента

## JSONConverter

Принимает и обрабатывает данные от сервиса учёта контингента.

Поля

|  |  |  |
| --- | --- | --- |
| \_serializer | IJsonSerializer | вспомогательный объект для работы с JSON |
| \_peopleURL | string | адрес доступа к персонам |
| \_groupURL | string | адрес доступа к группам |

Методы

|  |  |
| --- | --- |
| JSONConverter(string peopleURL, string groupURL) | конструктор по умолчанию (использует адреса доступа к сущностям) |
| GetObject<T>(string JSON) | получает объект указанного типа из строки JSON |
| GetObjectFromURL<T>(string URL) | получает объект указанного типа по заданному адресу |
| IsStudent(string ID) | определяет, является ли пользователь с данным ID студентом или нет |
| GetStudentFromID(string ID) | возвращает информацию о студенте по ID |
| GetGroupFromID(int ID) | возвращает информацию о группе по ID |
| GetTeacherFromID(string ID) | возвращает информацию о преподавателе по ID |
| GetGroupsList() | возвращает список групп |
| GetFirstCourseGroupList() | возвращает список групп первого курса |

## Teacher

Модель записи о преподавателе

Поля

|  |  |  |
| --- | --- | --- |
| sn | string | фамилия |
| givenName | string | имя |
| initials | string | отчество |
| cn | string | логин |
| title | string | статус |
| uid | string | ID |
| displayName | string | ФИО |
| isActive | bool | активность |
| \_links | [Links](#_Links) | информационные ссылки |

### Links

Информационные ссылки персоны

Поля

|  |  |  |
| --- | --- | --- |
| self | [Self](#_Self) | адрес пользователя |
| self | [Self](#_Self) | адрес профиля |
| groups | [Groups](#_Groups)[] | список групп |

### Self

Ссылки персоны

Поля

|  |  |  |
| --- | --- | --- |
| href | string | адрес ссылки |

### Groups

Информация о группах

Поля

|  |  |  |
| --- | --- | --- |
| id | string | ID группы |
| name | string | название группы |
| links | [GroupLink](#_GroupLink) | ссылка на группу |

### GroupLink

Ссылка на группу

Поля

|  |  |  |
| --- | --- | --- |
| self | string | адрес ссылки |

## Student

Модель записи о студенте

Поля

|  |  |  |
| --- | --- | --- |
| sn | string | фамилия |
| givenName | string | имя |
| mail | string[] | адреса электронной почты |
| uid | string | ID |
| displayName | string | ФИО |
| isActive | bool | активность |
| links | [Links](#_Links) | информационные ссылки |

## Group

Модель записи о группе

Поля

|  |  |  |
| --- | --- | --- |
| id | int | ID |
| name | string | название |
| curatorUid | string | ID куратора |
| headUid | string | ID старосты |
| finishedEducation | bool | закончено ли обучение |
| type | string | тип группы |
| \_embedded | [Embedded](#_Embedded) | вложение (список студентов) |
| curator | string[] | список кураторов |
| head | string[] | список старост |

### Embedded

Модель вложения со списком студентов

Поля

|  |  |  |
| --- | --- | --- |
| students | [Student](#_Student)[] | список студентов |

## GroupList

Модель списка групп

Поля

|  |  |  |
| --- | --- | --- |
| \_embedded | [Embedded](#_Embedded) | вложение (список групп) |

### Embedded

Модель вложения со списком групп

Поля

|  |  |  |
| --- | --- | --- |
| groups | [Group](#_Group) [] | список групп |

# Модуль работы с БД

Используется БД MySQL и ORM Dapper

## GroupSettingHelper

Класс управления настройками соответствия групп и шаблонов

Поля

|  |  |  |
| --- | --- | --- |
| connectionString | string | строка подключения к БД |

Методы

|  |  |
| --- | --- |
| GroupSettingHelper(string connectionString) | конструктор (необходимо задать строку подключения) |
| DeleteRecord(int Id\_template) | удаление записи из БД |
| GetRecord(int Id\_template) | получение настроек по ID шаблона |
| GetTeachRecord(string Id\_teacher) | получение преподавательских настроек по ID |
| GetTeachRecord(string Id\_teacher, int Id\_template) | получение настроек по ID преподавателя и шаблона |
| InsertRecord(List<[TemplateSetting](#_TemplateSetting)> records) | вставка записи в БД |

## TemplateSetting

Настройки шаблонов

Поля

|  |  |  |
| --- | --- | --- |
| Id\_template | int | ID шаблона |
| Id\_setting | int | ID настройки |

## TeachSetting

Настройки преподавателя

Поля

|  |  |  |
| --- | --- | --- |
| Id\_teach\_setting | int | ID настройки |
| Id\_teacher | string | ID преподавателя |
| Id\_group | int | ID группы |

## SettingHelper

Класс управления настройками

Поля

|  |  |  |
| --- | --- | --- |
| connectionString | string | строка подключения к БД |

Методы

|  |  |
| --- | --- |
| SettingHelper(string connectionString) | конструктор (необходимо задать строку подключения) |
| DeleteSettings(string Id\_teacher) | удаление настроек преподавателя по ID |
| GetSettings(string Id\_teacher) | получение настроек преподавателя по ID |
| InsertSettings(List<[TeachSetting](#_TeachSetting)> settings) | вставка настроек |

# Модуль работы с заданиями

## StudentTask

Параметры индивидуального задания

Поля

|  |  |  |
| --- | --- | --- |
| Id\_task | int | ID задания |
| Id\_student | string | ID студента |
| Id\_template | int | ID шаблона |
| Id\_group | int | ID группы |
| solve | string | название LaTeX-файла задания |
| maket | string | название LaTeX-файла решения |
| date | DateTime | дата выдачи задания |

## Template

Параметры шаблона

Поля

|  |  |  |
| --- | --- | --- |
| Id\_template | int | ID шаблона |
| Id\_teacher | string | ID преподавателя |
| template | string | файл шаблона (xml) |
| title | string | название шаблона |
| Id\_type | int | ID типа задания (формы контроля) |
| number | int | номер шаблона |

## TemplateRep

Класс управления заданиями

Поля

|  |  |  |
| --- | --- | --- |
| connectionString | string | строка подключения к БД |

Методы

|  |  |
| --- | --- |
| GetTemplates() | возвращает все шаблоны |
| CreateTemplate([Template](#_Template_1) template) | возвращает шаблон по указанному объекту шаблона |
| GetGroupTemplates(int id) | возвращает шаблоны по ID группы |
| GetTemplates(string id) | возвращает шаблоны по ID преподавателя |
| UpdateTemplate([Template](#_Template_1) template) | обновление шаблона в БД |
| GetTemplateFromID(int id) | возвращает шаблон по ID шаблона |
| Delete(int Id\_template) | удаляет шаблон по ID шаблона |
| GetTaskFromTemplate(int Id\_template) | возвращает список названий файлов заданий по ID шаблона |
| GetSolveFromTemplate(int Id\_template) | возвращает список названий файлов решений по ID шаблона |
| GetOldTask() | возвращает список заданий прошедших учебных лет |
| DeleteOldTask() | удаляет задания прошедших учебных лет |
| GetYearStart() | возвращает дату начала текущего учебного года |

## TemplateRep

Класс доступа к типам заданий

Поля

|  |  |  |
| --- | --- | --- |
| connectionString | string | строка подключения к БД |

Методы

|  |  |
| --- | --- |
| TempTypeRep(string connectionString) | конструктор |
| GetTypes() | возвращает список типов заданий |
| GetFromID(int id) | возвращает тип задания по ID |

## TemplateType

Параметры типа задания

Поля

|  |  |  |
| --- | --- | --- |
| Id\_type | int | ID |
| title | string | название |

## PathGenerator

Класс, генерирующий адреса файлов и папок

Поля

|  |  |  |
| --- | --- | --- |
| tempPath | string | каталог временных файлов |
| templatePath | string | каталог шаблонов |
| pdfPath | string | каталог pdf-документов |
| maketPath | string | каталог заданий |
| solvePath | string | каталог решений |
| pdfFile | string | путь к pdf-файлу задания |
| virtualTaskFile | string | путь к pdf-файлу задания |
| virtualSolveFile | string | путь к pdf-файлу решения |

Методы

|  |  |
| --- | --- |
| PathGenerator(string template, string maket, string solve, string templateFolder, string maketFolder, string solveFolder, string pdFolder, string tempFolder, string rootPath) | конструктор, генерирующий адреса |

## StudTemplateTree

Дерево заданий студента

Поля

|  |  |  |
| --- | --- | --- |
| Nodes | List<[Node](#_Node_2)> | список вершин дерева |

Методы

|  |  |
| --- | --- |
| StudTemplateTree(string id, string DBCOnnet, [JSONConverter](#_JSONConverter) converter) | конструктор, строит дерево заданий |

### Node

Вершина дерева заданий студента

Поля

|  |  |  |
| --- | --- | --- |
| type | [TemplateType](#_TemplateType) | тип задания |
| templates | List<[Template](#_Template_1)> | список заданий |

Методы

|  |  |
| --- | --- |
| Node() | конструктор |

## TaskSender

Класс получения заданий (*отправления* с точки зрения сервера)

Поля

|  |  |  |
| --- | --- | --- |
| \_connectionString | string | строка соединения с БД |
| task | StudentTask | задание |
| \_converter | [JSONConverter](#_JSONConverter) | экземпляр класса работы с сервисом учёта контингента |

Методы

|  |  |
| --- | --- |
| TaskSender(string connectionString, [JSONConverter](#_JSONConverter)converter) | конструктор |
| TaskSend(string Id\_student, int Id\_template) | получает задание по ID задания и ID студента |

## TaskTree

Дерево заданий преподавателя

Поля

|  |  |  |
| --- | --- | --- |
| \_connectionString | string | строка соединения с БД |
| types | List<[TemplateType](#_TemplateType_1)> | список типов заданий |
| records | List<Record> | список заданий |

Методы

|  |  |
| --- | --- |
| TaskTree(string id, string connectionString) | конструктор, строит дерево заданий |

### TemplateType

Вершина-тип задания

Поля

|  |  |  |
| --- | --- | --- |
| Id\_type | int | ID типа задания |
| type | string | название типа задания |
| templates | List<[Template](#_Template_2)> | список заданий |

#### Template

Вершина- шаблон

Поля

|  |  |  |
| --- | --- | --- |
| Id\_template | int | ID шаблона |
| title | string | название шаблона |
| number | int | номер шаблона |
| groups | List<[Group](#_Group_1)> | список групп, получивших задание по шаблону |

##### Group

Вершина-группа

Поля

|  |  |  |
| --- | --- | --- |
| Id\_group | int | ID группы |
| studCount | int | количество студентов |
| title | string | название группы |
| students | List<[Student](#_Student_1)> | список студентов группы, получивших задание |

###### Student

Вершина-студент

Поля

|  |  |  |
| --- | --- | --- |
| Id\_student | string | ID студента |
| fio | string | ФИО студента |
| task | [StudTask](#_StudTask) | задание |

StudTask

Вершина-задание

Поля

|  |  |  |
| --- | --- | --- |
| Id\_task | int | ID задания |
| solve | string | путь к файлу решения |
| maket | string | путь к файлу задания |

### Record

Класс, содержащий необходимую информацию о задании

Поля

|  |  |  |
| --- | --- | --- |
| Id\_task | int | ID задания |
| Id\_student | string | ID студента |
| Id\_group | int | ID группы |
| Id\_template | int | ID шаблона |
| Id\_type | int | ID типа задания |
| typeTitle | string | название типа задания |
| templateTitle | string | название шаблона задания |
| number | int | номер задания |
| solve | string | путь к файлу решения |
| maket | string | путь к файлу задания |
| issue\_date | DateTime | дата получения задания |

## TeachTemplateTree

Дерево шаблонов преподавателя

Поля

|  |  |  |
| --- | --- | --- |
| Nodes | List<[Node](#_Node_2)> | список вершин дерева |

Методы

|  |  |
| --- | --- |
| TeachTemplateTree (string id, string DBCOnnet, [JSONConverter](#_JSONConverter) converter) | конструктор, строит дерево заданий |

# Модуль работы с токеном JWT

## TokenDecoder

Класс расшифровки токена JWT

Методы

|  |  |
| --- | --- |
| Decode(string tokenString, string secret = "") | расшифровывает токен JWT |

## Token

Модель токена JWT

Поля

|  |  |  |
| --- | --- | --- |
| iss | string | пространство имен Access Control, выдавшего токен |
| sub | string | содержимое токена |
| aud | string | значение области, которое используется для запроса токена |
| iat | string | время, когда маркер был выдан и может быть использован для определения возраста JWT |
| exp | string | дата, когда истекает срок действия маркера |

# Модули сайта

## Logger

Класс, отвечающий за ведение логов работы сервера

Методы

|  |  |
| --- | --- |
| Logger() | конструктор |
| LogWrite(Exception exp) | записывает подробный отчёт об исключении |
| LogWrite(string msg) | записывает текст сообщения (например исключения) |

## Менеджер расписания задач

### Clearer

Класс удалялки файлов, реализующий интерфейс IJob

Методы

|  |  |
| --- | --- |
| Execute(IJobExecutionContext context) | удаляет устаревшие файлы |

### FileClearManager

Менеджер удалялки файлов

Методы

|  |  |
| --- | --- |
| Start() | запускает удалялку файлов по расписанию (каждые 30 дней) |

Устаревшими считаются файлы, созданные ранее ближайшего прошедшего первого сентября

## Пользовательские компоненты

### tabs

Панель вкладок преподавателя

### user\_name

Панель пользователя (ФИО, кнопки справки и выхода)

### user\_panel

Панель пользователя (ФИО, кнопки справки и выхода)

### cook

Модуль работы с cookie. Проверяет авторизацию пользователя.

# Обзор типового задания

Одной из основных функций разрабатываемого веб-приложения является генерация индивидуального задания по разделу «Матричные вычисления» дисциплины «Информатика». Типовое задание включает в себя исходные данные и перечень операций, которые необходимо выполнить. К исходным данным относится перечень матриц. Над ними возможно выполнение следующих операций:

* вычисление матрицы по формуле;
* вычисление определителя;
* нахождение обратной матрицы;
* транспонирование матрицы;
* нахождение элемента матрицы;
* вычисление значения по элементам матрицы.

Кроме того, в одном из заданий коэффициенты вычисленной по формуле матрицы используются при решении системы уравнений. К задаче вычисления матрицы по формуле предъявляются особые требования. Формула должна соответствовать нескольким ограничениям:

* количество исходных матриц;
* размерность итоговой матрицы;
* наличие числа;
* наличие вектор-строки и вектор-столбца;
* наличие квадратной матрицы;
* вычислимость.

Желательно, чтобы формулы не повторялись и не были сильно похожи друг на друга. Также коэффициенты матриц должны быть не слишком большими или маленькими, что упрощает работу как студентам, так и преподавателям.

Среди перечисленных задач вычисление матрицы по формуле является наиболее сложной и требовательной операцией в плане алгоритмизации. Таким образом, возникает потребность в дополнительных исследованиях, направленных на поиск алгоритма генерации формулы с заданными требованиями.

Стоит отметить, что в научной литературе вопрос генерации формул для вычисления матриц должным образом не освещён. Присутствуют работы, посвящённые выполнению отдельных действий. В монографии В. В. Кручинина «Генераторы в компьютерных учебных программах» [1] представлены алгоритмы заданий по нахождению определителя, решению системы линейных уравнений, проверке умножения матриц на допустимость. В публикации Я. Ю. Коновалова «Генератор контрольных заданий по высшей математике: опыт создания и применения» [2] рассмотрена генерация матричного уравнения вида , и , а также задач на решение систем линейных уравнений. Поскольку не найдено готовых алгоритмов генерации формул, решено произвести собственное исследование по этому вопросу.

Таким образом, целью исследования является разработка алгоритма генерации формул для вычисления матриц с учётом ограничений по следующим параметрам:

* количество исходных матриц;
* размерность итоговой матрицы;
* наличие числа;
* наличие вектор-строки и вектор-столбца;
* наличие квадратной матрицы;
* вычислимость.

# Алгоритмы генерации формулы

## Алгоритм генерации формулы (версия №1)

Генерация формулы осуществляется как рекурсивная генерация текстовой строки, к которой на каждом шаге добавляется операция, матрица или числа. Генерация завершается, когда добавлено необходимое количество операций. Изначально заданы названия матрицы и допустимые операции. Принцип основан на том, что формулу можно представить, как последовательность вида



Рис. 7. Общее представление формулы

Случайным образом генерируются такие пары и объединяются в текстовую строку, которая и будет искомой формулой.

Случайным образом к строке дописывается либо название матрицы из готового списка, либо число, которое также является случайным целым в диапазоне от 1 до 10. Если выбрана матрица, то она удаляется из списка, чтобы избежать повторений. Если список матриц пуст, то добавляются только числа. Затем случайным образом определяется тип (унарная или бинарная) и конкретная операция, которая добавляется к формуле. Если выбрана бинарная операция, то также добавляется матрица или число. Схематично по шагам этот процесс представлен на рис. 8.



Рис. 8. Общее представление формулы

Процесс повторяется, пока не будет создано необходимое количество операций.



Рис. 9. Генерация формулы

Следующий этап – расстановка слагаемых по уровням иерархии. Элементы каждого дополнительного уровня обрамляются скобками.



Рис. 9. Расстановка уровней и скобок

Очевидно, что такая организация формулы имитирует дерево, причём создаёт дополнительные трудности из-за того, что вся информация хранится в текстовой строке, которая исключает какую-либо иерархию. Поэтому в одной из следующих реализаций алгоритма было решено генерировать не непосредственно строковое представление формулы, а соответствующее дерево.

Данный алгоритм не гарантирует, что в формуле будет содержаться нужное количество матриц, что для полученной формулы можно будет сгенерировать матрицы, удовлетворяющие ограничениям по размерности, и что вычисление матрицы по такой формуле в принципе возможно.

## Алгоритм генерации формулы (версия №2)

В данном алгоритме решается исключительно задача генерации формулы для расчёта матрицы, т.е. генерация матриц или нахождения решения в нём не реализованы. Содержимое формулы определяется тремя параметрами: количество матриц, операций и скобок.

Задача нахождения формулы интерпретирована как задача поиска в глубину. При этом заданы:

* начальное состояние;
* порождающая функция;
* условия прекращения поиска.

Дерево поиска содержит вершины сгенерированных состояний. В состоянии фиксируется родительское состояние, список дочерних состояний, запись формулы, количество действий и скобок. Помимо этого, в дереве хранится список матриц, операций, скобок и чисел.

Начальное состояние не имеет не родителя, ни потомков, а количество операций и скобок равно нулю. Поиск завершается, если в текущем состоянии расставлены все скобки и операции.

Порождающая функция максимально упрощена. К текущему состоянию прибавляется каждая матрица, операция, скобка и число. Таким образом, количество дочерних состояний составляет около тридцати.



Рис. 9. Состояние поиска и список его потомков

Каждое дочернее состояние проходит процедуру проверки. Для этого оно преобразуется в строковое представление, в котором используются специальные обозначения для следующих категорий:

* матрица (M);
* бинарная операция (B);
* унарная операция (U);
* число (N).

Скобки остаются без изменений. Проверка заключается в поиске запрещённых последовательностей, которые соответствуют недопустимым комбинациям объектов формулы. Если такая последовательность встречается в текстовом представлении, то соответствующее состояние не добавляется в дерево поиска. Ниже представлены некоторые запрещённые последовательности:

* MM – две матрицы подряд;
* (O – операция сразу после открывающей скобки;
* MN – число сразу после матрицы (без операции);
* (N) – число, заключённое в скобки.

Также недопустимо, чтобы формула начиналась с закрывающей скобки или операции.

Алгоритм непосредственно поиска довольно прост:

1. Начальное состояние добавляется в список открытых вершин.
2. Выбирается первая вершина из списка открытых вершин и происходит её «раскрытие» с помощью порождающей функции, после чего дочерние состояния заносятся в список, а родительская вершина – удаляется.
3. Предыдущий пункт повторяется до тех пор, пока не будет найдено конечное состояние и поиск будет прекращён.

В данном случае нет необходимости хранить уже раскрытые вершины, так как повторение состояний исключено.

Основная сложности данного алгоритма заключается в фильтрации дочерних состояний, т.к. необходимо заранее определить все недопустимые комбинации элементов. Особенно усложнена проверка элементов, распложенных на разных уровнях иерархии, а также корректности расстановки скобок. К тому же, практически не устранены недостатки, присущие предыдущей версии алгоритма.

## Алгоритм генерации формулы (версия №3)

Данный алгоритм позволяет создавать формулу нахождения матрицы требуемой размерности с указанным количеством действий, матриц, чисел. Происходит генерация формулы и входящих в неё матриц и чисел. Основная идея заключается в том, что операциям соответствуют матрицы, которые получаются в результате их вычисления. Поэтому операции имеют те же атрибуты, что и матрицы, и их можно преобразовывать друг в друга. Таким образом, можно определить итоговую матрицу и преобразовать её в дерево операций над матрицами.

Алгоритм проходит через приведённые ниже этапы.

1. Генерация дерево вычисления формулы.

В начале работы алгоритма создаётся вершина, соответствующая матрице, которая будет получена в результате вычислений по формуле. Затем составляется список матриц, и происходит раскрытие случайно-выбранной матрицы из списка. Процесс повторяется столько раз, сколько операций необходимо создать.

При раскрытии матрицы её вершина заменяется на вершину-операцию. Для замены случайным образом выбирается тип операции: бинарная или унарная.

Если выбрана бинарная:

* если родительская вершина – тоже бинарная операция, то будет создана вершина с другой операцией (родитель – «\*», потомок – «+» и наоборот);
* если родительская вершина – операция произведения, то дочерняя операция суммы берётся в скобки;
* если родительская вершина – унарная операция или пустая вершина (т.е. корень дерева), то будет создана бинарная операция (произведение или сложение – выбирается случайным образом, но для умножения вероятность 0.6) и берётся в скобки.

Корневая вершина заменяется только на бинарную операцию.

Если выбрана унарная операция, то вершине добавляются фигурные скобки, которые необходимы для вывода в LaTeX, и:

* если родительская вершина – тоже унарная операция, то
  + вершине назначаются круглые скобки;
  + если родительская вершина –операция транспонирования, и если матрица квадратная, то добавляется операция нахождения обратной матрицы, в противном случае добавляется бинарная операция;
  + если родительская вершина – операция нахождения обратной матрицы, то добавляется операция транспонирования;
* если родительская вершина – бинарная операция или пустая вершина и матрица квадратная, то добавляется операция нахождения обратной матрицы, в противном случае – операция транспонирования.

Сразу же добавляются дочерние вершины-матрицы: две для бинарной операции и одна для унарной. Сразу же назначаются размерности вершин в зависимости от операции:

* умножение – количество строк первой вершины равно количеству строк текущей, а количество столбцов второй – количеству столбцов текущей вершины; количество столбцов первой матрицы равно количеству строк второй, и определяется как максимальное целое число, не равное размерностям текущей матрицы и не превышающее максимально-допустимую размерность;
* сложение – размерности дочерних вершин совпадают с родительскими;
* транспонирование – количество строк дочерней вершины соответствует числу столбцов текущей вершины, а количество столбцов – количеству строк;
* нахождение обратной матрицы –размерности дочерней вершины соответствуют размерностям текущей.

На рис. 6 представлен процесс генерации дерева вместе с вычислением размерностей в случае, если по условию задано 3 операции.



Рис. 6. Состояние поиска и список его потомков

1. Подгонка полученного количества матриц и чисел под необходимое.

Количество сгенерированных матриц (на данном этапе числа тоже считаются матрицами) может отличаться от требуемого, поэтому осуществляется его подгонка. Возможны две ситуации: матриц недостаточно либо их слишком много. При недостатке матриц происходит добавление матриц по одной до тех пор, пока количество матриц не совпадёт с необходимым. Для этого необходимо преобразовать унарную операцию в бинарную, добавив таким образом в дерево одну матрицу, либо убрать унарную операцию и добавить бинарную. Чтобы убрать матрицу, необходимо преобразовать бинарную операцию в унарную, либо удалить бинарную операцию и добавить унарную. Эти действия применимы, если операции имеют в качестве хотя бы одного из своих операндов матрицу. В противном случае изменения затронут слишком большое количество матриц, и результат станет трудно контролируемым. На рис. 7 представлены варианты изменения дерева при добавлении и удалении матриц.



Рис. 7. Подгонка количества матриц

Зависимость количества матриц (*x*) от количества операций (*y*) описывается следующим образом:

В завершении этапа происходит генерация вершин-чисел. Для этого случайным образом выбирается бинарная операция, и одна из матриц-операндов преобразуется в число. Процесс повторяется столько раз, сколько чисел необходимо сгенерировать.

Если по заданию необходимо сгенерировать вектор-строку и вектор-столбец, то проверяется, есть ли на нижнем уровне операция умножения (т.е. умножение непосредственно матриц). Если есть, то количество столбцов первой матрицы и количество строк второй устанавливаются равными единице. Если же такой операции не оказалось, то производится поиск операции сложения нижнего уровня, и найденная операция преобразуется в операцию умножения матриц с описанными ранее размерностями. Если и такой операции не нашлось, то в операцию умножения преобразуется унарная операция.

1. Генерация, именование матриц и чисел.

Матрицы заносятся в список, и для каждой матрицы выбирается название, соответствующее заглавной букве английского алфавита с номером, равным номеру матрицы в списке. Числа заносятся в отдельный список и им случайным образом присваиваются значения от 2 до 10.

В этом алгоритме исправлены некоторые недостатки предыдущих версий, но часть из них сохранилась. В частности, нельзя гарантировать, что по заданным ограничениям можно сгенерировать формулу, т.к. не удалось вывести критерий допустимости, связывающий все исходные параметры.

## Алгоритм генерации формулы (версия №4)

Данный алгоритм позволяет создавать формулу нахождения матрицы требуемой размерности с указанным количеством матриц и чисел, а также с наличием специальных матриц (вектор-строка/столбец, квадратная матрица) и операций (инвертирование матрицы с нулевым определителем). Происходит генерация формулы и входящих в неё матриц и чисел. Основная идея заключается в том, что можно изначально определить набор матриц и чисел, а затем создавать операции, объединяющие их, и в конечном итоге прийти к тому, всё сойдётся к одной операции, результат вычисления которой будет соответствовать итоговой матрице.

Процесс генерации дерева проходит через следующие этапы:

1. Генерация вершин-матриц и вершин-чисел. Сначала генерируется необходимое количество вершин-матриц и вершин-чисел. Затем некоторые вершины-матрицы преобразуются в вершины-квадратные матрицы и вершины-вектора в соответствии с заданием. Если изначально задана операция на нахождение обратной матрицы для квадратной матрицы с нулевым определителем, то одна матрица преобразуется в квадратную и ей назначается нулевой определитель, а также добавляется операция нахождения обратной матрицы. В случае, когда необходимо сгенерировать вектор-строку и вектор-столбец, для соответствующих матриц применяется операция умножения (рис. 1).



Рис. 1. Дерево после генерации вершин-матриц и вершин-чисел

1. Генерация дерева вершин. Генерация происходит «снизу-вверх», т.е. от исходных вершин к вычисляемой. Случайным образом выбирается две вершины верхнего уровня и тип матричной операции. Оптимальная вероятность выбора бинарной операции составила 80%, что обеспечивает наличие небольшого числа унарных операций. Конкретная операция также выбирается случайным образом, при этом вероятность выбора умножения – 60% и сложения – 40%, а операции транспонирования и нахождения обратной матрицы равновероятны (по 50%). Вершина, соответствующая добавленной операции, заносится в список вершин верхнего уровня, а её операнды удаляются из списка. Процесс повторяется до тех пор, пока на верхнем уровне не останется одна вершина, соответствующая заключительной операции нахождения матрицы по формуле (рис. 2).



Рис.2. Процесс генерации дерева

1. Корневой вершине присваивается требуемая размерность. Таким образом гарантируется, что в результате выполнения всех операций получится необходимая матрица.
2. «Свёртка» дерева: вместо двух унарных операций, следующих одна за другой, оставляется одна; если применяются две одинаковые бинарные операции подряд, то одна из них меняется на другую операцию (рис. 3).



Рис. 3. Операция свёртки

1. Расстановка скобок. Для текстового отображения формулы необходимо расставить круглые скобки. Для этого производится рекурсивный обход дерева и анализ операций вершин и их потомков. Текущая операция заключается в скобки если:
   1. она является бинарной операцией, а родительская – унарной;
   2. она является операцией сложения, а родительская – операцией умножения.

На рис. 4 представлено дерево с расставленными скобками.



Рис. 4. Дерево с расставленными скобками

1. Именование матриц и определение чисел. Для текстового вывода формулы необходимо задать матрицам соответствующие обозначения. Для этого матрицам по порядку назначаются буквы английского алфавита, а корневой вершине присваивается следующая по прядку буква. Числа перебираются по порядку и им присваиваются случайные числа в выбранном диапазоне.



Рис. 5. Дерево с поименованными матрицами и сгенерированными числами

1. Расстановка размерностей. Каждой ветви дерева соответствует матрица: вершина-матрица уже включает её в себя, а вершина-операция подразумевает, что матрица является результатом её выполнения. Задача данного этапа – определить размерности вершин-матриц, исходя из размерности корневой вершины, которая задана по условию. При этом следует учитывать, что некоторые матрицы уже содержат некоторые сведения о размерностях, которые накладывают дополнительные ограничения: у вершины-квадратной матрицы должно совпадать количество строк и столбцов, а вершина-вектор-строка должна иметь только одну строку. Матричные операции также предъявляют специфические требования:

* для умножения матриц необходимо, чтобы количество столбцов первой матрицы совпадало с количеством строк второй;
* для сложения матриц необходимо, чтобы они имели совпадающие размерности;
* операция нахождения обратной матрицы применима только к квадратной матрице.

При этом сложение и умножение возможны для любых матрицы и числа, а транспонировать можно любую матрицу. Кроме того, желательно, чтобы матрицы имели размерности, близкие к максимальной (указанной в требованиях), иначе задание получится недостаточно сложным. В то же время нужно не допускать появления нежелательных квадратных матриц. Таким образом, основная сложность данного этапа заключается в соблюдении ограничений, накладываемых требованиями к заданию и спецификой матричных операций.

Расстановка размерностей начинается с корневой вершины, параметры которой заданы изначально. Рассматриваются дочерние вершины, и их размерности определяются по следующим принципам:

* если текущая вершина – это операция умножения и один из её потомков – вершина-число, то размерности другой дочерней вершины соответствуют размерностям текущей;
* если текущая вершина – это операция умножения и ни один из её потомков не является вершиной-числом, то количество строк первой вершины равно количеству строк текущей, а количество столбцов второй – количеству столбцов текущей вершины; количество столбцов первой матрицы равно количеству строк второй, и определяется как максимальное целое число, не равное размерностям текущей матрицы и не превышающее максимально-допустимую размерность;
* если текущая вершина – это операция сложения, то размерности дочерних вершин совпадают с родительскими;
* если текущая вершина – это операция транспонирования, то количество строк дочерней вершины соответствует числу столбцов текущей вершины, а количество столбцов – количеству строк;
* если текущая вершина – это операция нахождения обратной матрицы, то размерности дочерней вершины соответствуют размерностям текущей.



Рис. 6. Дерево с расставленными размерностями

1. Проверка требований. На заключительном этапе генерации дерева проверяется соблюдение требований, накладываемых заданием, таких как:
   1. наличие квадратных матрицы;
   2. наличие вектор-строки и вектор-столбца;
   3. наличие операции нахождения обратной матрицы;

После этого происходит заполнения матриц с учётом их размерностей и допустимости дробных значений. В случае, когда нужно обеспечить равенство определителя нулю, происходит случайный выбор типа подмножества матрицы (строка или столбец) и двух его случайных номеров. Элементы одного выбранного подмножества заменяются на элементы другого, умноженные на число от 2 до 4.

Затем производится вычисление итоговой матрицы. Для этого производится рекурсивное выполнение операций над элементами дерева. Рекурсивное углубление происходит до тех пор, пока операндами не окажутся матрицы и числа, после чего происходит вычисление результата операции, который заносится в матрицу, соответствующую операции. Таким образом, корневая операция будет содержать в себе итоговую матрицу.

Нахождение результата позволяет эффективно отсеять невычислимые формулы. Кроме того, коэффициенты полученной матрицы проверяются на принадлежность допустимому интервалу (от 0 до 1000).

Процесс генерации дерева для формулы нахождения матрицы повторяется до тех пор, пока не будут выполнены все предъявляемые к ней требования. Несмотря на то, что для создания индивидуального задания приходится многократно повторять процедуру генерацию дерева, процесс происходит достаточно быстро, потому что генерация занимает очень мало времени.

Данный алгоритм решает основные поставленные перед ним задачи, но, тем не менее, по-прежнему не гарантирует нахождение вычислимой формулы. Выяснилось, что не каждый набор матриц может быть сведён в одну формулу, и не всегда удаётся корректно расставить размерности. Особенно серьёзные проблемы возникают при генерации формул из небольшого количества матриц, при условии, что они являются специальными (квадратная матрица, вектор-строка/вектора-столбец).

## Алгоритм генерации формулы (версия №4.1)

Данный алгоритм вносит небольшие изменения, призванные улучшить работу генератора с формулами, содержащими небольшое количество матриц, при условии, что они являются специальными (квадратная матрица, вектор-строка/вектора-столбец).

В случае, когда генератор сложно улучшить принципиально, можно найти частные решения для известных проблем. Т. о. на работу генератора влияют следующие параметры:

* количество матриц (от 2 до 5);
* наличие числа;
* наличие квадратной матрицы;
* наличие вектор-строки/столбца.

Возможно наличие сразу нескольких параметров. Для каждой возможной комбинации необходим свой шаблон. На рис. 14 представлена схема организации шаблонов, где V – это вектор-строка/столбец, а Q – квадратная матрица.



Рис. 14. Схема организации шаблонов.

Каждый шаблон включает в себя базовую структуру дерева. На рис. 14 представлены некоторые шаблоны, где V – это вектор-строка/столбец, Q – квадратная матрица, N – число, B – бинарная операция.



Рис. 14. Схема организации шаблонов: а – шаблон №1; б – шаблон №20, в – шаблон №21

Для бинарных операций возможна перестановка операндов местами с соответствующим изменением размерностей. Кроме того, возможно добавление унарных операций.

Данный алгоритм позволяет гарантированно решить проблемы, связанные с обработкой частных случаев, но обладает недостатками:

* генерируемые формулы слабо отличаются друг от друга;
* необходимо разработать достаточно большое количество шаблонов;
* изменения алгоритма могут привести к неработоспособности шаблонов и необходимости их переделывания.

Поэтому было решено создать такой алгоритм, который позволял бы автоматически (без шаблонов) создавать формулы при любых исходных параметрах.

## Алгоритм генерации формулы (версия №5)

Данный алгоритм – это комбинация предыдущих алгоритмов (версии 3 и 4) с некоторыми доработками. Основная идея заключается в том, что из конечного состояния можно сгенерировать необходимое количество промежуточных матриц, а затем преобразовать их таким образом, чтобы гарантированно соблюсти требуемые ограничения.

Очевидно, что любую матрицу можно представить в виде произведения вектор-столбца и вектор строки (именно в таком порядке), или квадратной матрицы и матрицы общего вида (т.е. матрицы, которая не является каким-либо частным случаем). На рис. представлено, каким образом происходит такое преобразование. Исключением являются матрицы-вектора, потому что в таком случае одна из производных матриц вырождается в матрицу из одного элемента, что нежелательно по требованиям задачи.



Рис. 14. Получение вектор-строки/столбца и квадратной матрицы из произвольной матрицы

Таким образом, для генерации как вектор-строки и столбца, так и квадратной матрицы, необходимо добавить в дерево вычисления формулы промежуточную матрицу, которая в дальнейшем будет преобразована в необходимые матрицы.

Необходимо учитывать, что по заданию может требоваться генерация матриц-векторов, квадратной матрицы и числа. В случае, если требуется генерация как матриц-векторов, так и квадратной матрицы, сначала нужно сгенерировать квадратную матрицу, а затем преобразовать матрицу общего вида в произведение матриц-векторов, потому что в противном случае образуется квадратная матрица из одного элемента, что недопустимо. Таким образом будет создано минимальное количество матриц для такого набора параметров (3 матрицы). Если кроме того необходимо сгенерировать ещё и число, то любую из матриц на любом этапе можно представить, как произведение матрицы на число или как их сумму. В этом случае также не происходит образование дополнительных матриц, поэтому можно считать, что наличие числа не накладывает дополнительных ограничений на формулу.



Рис. 14. Генерация формул с разными ограничениями из произвольной матрицы

Таким образом, можно вывести формулу для нахождения количества матриц общего вида, которых нужно сгенерировать, а также минимального количества матриц по заданию:

где – количество матрицы общего вида; – количество матриц по заданию; – количество пар матриц-векторов; – количество квадратных матриц.

При генерации формул с количеством матриц, превышающим минимальное, для преобразования могут использоваться разные матрицы, выбираемые случайным образом.

Алгоритм генерации формулы включает приведённые ниже этапы.

1. Создание начальной вершины, соответствующей итоговой матрице.
2. «Разворачивание» начальной матрицы в дерево с необходимым количеством матриц общего вида. Происходит выбор случайной вершины-матрицы и преобразование её в вершину-операцию. Процесс происходит также, как и в алгоритме версии 3. Кроме того, осуществляется расстановка скобок, как в алгоритме версии 4.
3. Добавление к корневой вершине унарной операции. Если итоговая матрица неквадратная, то допустимо только транспонирование, в противном случае – ещё и нахождение обратной матрицы. Операция добавляется с вероятностью в 40%.
4. Добавление числа – выбирается случайная матрица и преобразуется в бинарную операцию, один из операндов которой преобразуется в число.
5. Добавление квадратной матрицы – выбирается случайная матрица и преобразуется в произведение квадратной матрицы и матрицы общего вида.
6. Добавление матриц-векторов – выбирается случайная матрица и преобразуется в произведение матриц-векторов.
7. Именование матриц и определение чисел (как в алгоритме версии 4).
8. Расстановка размерностей матриц (как в алгоритме версии 4).
9. Заполнение матриц (как в алгоритме версии 4).
10. Вычисление итоговой формулы (как в алгоритме версии 4).
11. Проверка ограничений (как в алгоритме версии 4).

Пункт №4 может быть выполнен как после 3-го, так и после 4-го или 5-го пункта, что определяется случайным образом. Если выполняются не все ограничения (пункт 11), то процесс генерации повторяется заново.

В результате получается формула для вычисления матрицы, набор исходных матриц и матрица-результат. За счёт того, что в процессе генерации учитывается специфика матричных операций, практически всегда генерируется вычислимая формула, но результат часто не удовлетворяет требованиям. Например, получаются слишком большие или слишком маленькие коэффициенты, что усложняет ручную проверку работ. Благодаря тому, генерация формулы происходит очень быстро, возможно повторение процесса до тех пор, пока не будет получен удовлетворительный результат. Алгоритм проверен на всём множестве допустимых параметров генерации и выдаёт удовлетворительные результаты. Генерируемые задания в достаточной степени отличаются друг от друга и выглядят принципиально-разными.

# Выводы

В таблице 1 представлено сравнение разработанных алгоритмов по реализованным возможностям.

*Таблица**1*

Сравнение разработанных алгоритмов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Характеристика | Версия алгоритма | | | | | |
| 1 | 2 | 3 | 4 | 4.5 | 5 |
| генерация формулы | + | + | + | + | + | + |
| расстановка скобок | +/- | + | + | + | + | + |
| контроль размерности выходной матрицы | - | - | + | + | + | + |
| контроль количества чисел | - | - | + | + | + | + |
| контроль количества операций | + | + | + | - | - | - |
| контроль количества матриц | - | + | + | + | + | + |
| контроль количества скобок | - | + | - | - | - | - |
| гарантия генерации матриц специальных видов | - | - | - | + | + | + |
| фильтрация недопустимых значений элементов матриц | - | - | - | + | + | + |
| проверка вычислимости формулы | - | - | - | + | + | + |
| фильтрация несовместимых параметров задания | - | - | - | - | - | + |
| гарантия генерации вычислимой формулы | - | - | - | - | + | + |
| нешаблонность заданий | + | + | + | + | - | + |

Из сравнения результатов следует, что последняя (5-я) версия алгоритма обладает наибольшей функциональностью и соответствует всем наиболее важным требованиям. Данный алгоритм реализован в модуле разрабатываемого веб-приложения и применяется в процессе составления индивидуальных заданий по матричным операциям дисциплины «Информатика».

# Список литературы

1. Кручинин, В.В. Генераторы в компьютерных учебных программах / В.В. Кручинин. – Томск: Изд-во Том. ун-та, 2003. – 200 с.
2. Коновалов, Ю.Я. Генератор контрольных заданий по высшей математике: опыт создания и применения / Ю.Я. Коновалов, С.К. Соболев // Электронный научно-технический журнал «Инженерный вестник». – 2014. – №4. – С. 1046-1055.