

Федеральное государственное автономное образовательное учреждение
высшего образования "Национальный исследовательский Нижегородский
государственный университет им. Н.И. Лобачевского"

Отчёт №2

по учебной дисциплине
«Алгоритмы и структуры данных»

Студента Ципина Д.Д.

группы 3824Б1ФИ2

Нижний Новгород – 2025 г.

Постановка задачи

Цель данной работы — создания программных средств, поддерживающих эффективное хранение матриц специального вида (верхнетреугольных) и выполнение основных операций над ними: сложение, вычитание, копирование, сравнение.

Описание класса TVector

Класс TVector представляет собой шаблонный класс для хранения одномерного массива элементов произвольного типа. Он реализует базовые операции над векторами и поддерживает работу как с скалярами, так и с другими векторами.

Описание ключевых методов TVector

TVector(int s = 10, int si = 0)

- **Параметры:** s — размер вектора; si — индекс первого элемента.
- **Функционал:** выделяет память под массив длиной s, инициализирует элементы значениями по умолчанию.
- **Сложность:** O(n), где n — размер вектора.

TVector(const TVector& v)

- **Параметры:** v — другой вектор.
- **Функционал:** конструктор копирования, создаёт новый вектор, копируя все элементы.
- **Сложность:** O(n).

~TVector()

- **Функционал:** освобождает динамически выделенную память.

ValType& operator[](int pos)

- **Параметры:** pos — индекс элемента.
- **Функционал:** возвращает ссылку на элемент по индексу, с учётом StartIndex. При выходе за границы выбрасывает исключение.

bool operator==(const TVector& v) const

- **Функционал:** сравнивает два вектора на равенство (размер, индекс и элементы).
- **Сложность:** O(n).

TVector& operator=(const TVector& v)

- **Параметры:** v — другой вектор.
- **Функционал:** оператор присваивания, копирует все элементы и параметры.
- **Сложность:** O(n).

TVector operator+(const ValType& val)

- **Функционал:** прибавляет скаляр ко всем элементам.
- **Сложность:** O(n).

TVector operator-(const ValType& val)

- **Функционал:** вычитает скаляр из всех элементов.
- **Сложность:** O(n).

TVector operator*(const ValType& val)

- **Функционал:** умножает все элементы на скаляр.
- **Сложность:** O(n).

TVector operator+(const TVector& v)

- **Функционал:** поэлементное сложение двух векторов одинакового размера и индекса.
- **Сложность:** O(n).

TVector operator-(const TVector& v)

- **Функционал:** поэлементное вычитание.
- **Сложность:** $O(n)$.

ValType operator*(const TVector& v)

- **Функционал:** скалярное произведение двух векторов.
- **Сложность:** $O(n)$.

friend std::istream& operator>>(std::istream& in, TVector& v)

- **Функционал:** считывает элементы вектора из потока.
- **Сложность:** $O(n)$.

friend std::ostream& operator<<(std::ostream& out, const TVector& v)

- **Функционал:** выводит элементы вектора в поток, разделяя пробелами.
- **Сложность:** $O(n)$.

Описание класса TMatrix

Класс TMatrix представляет собой шаблонную структуру данных для хранения верхнетреугольной матрицы. Он построен на основе класса TVector, где каждая строка матрицы реализуется как вектор, начинающийся с определённого индекса. Класс поддерживает операции сравнения, присваивания, сложения и вычитания матриц одинакового размера, а также ввод и вывод в удобном табличном формате.

Описание ключевых методов TMatrix

TMatrix(int s = 10)

- **Параметры:** s — размер матрицы.
- **Функционал:** создаёт верхнетреугольную матрицу размера $s \times s$.

Каждый элемент матрицы хранится в векторе, при этом строки имеют разную длину (первая строка — s элементов, вторая — $s-1$, и т.д.).

- **Сложность:** $O(n^2)$, где n — размер матрицы.

TMatrix(const TMatrix& mt)

- **Параметры:** mt — другая матрица.

- **Функционал:** конструктор копирования, создаёт точную копию матрицы.
- **Сложность:** $O(n^2)$.

TMatrix(const TVector<TVector<ValType>>& mt)

- **Параметры:** mt — вектор векторов.
- **Функционал:** преобразует вектор векторов в матрицу.
- **Сложность:** $O(n^2)$.

bool operator==(const TMatrix& mt) const

- **Функционал:** сравнивает две матрицы на равенство (размер и все элементы).
- **Сложность:** $O(n^2)$.

bool operator!=(const TMatrix& mt) const

- **Функционал:** проверяет неравенство матриц.
- **Сложность:** $O(n^2)$.

TMatrix& operator=(const TMatrix& mt)

- **Параметры:** mt — другая матрица.
- **Функционал:** оператор присваивания, копирует все элементы и параметры.
- **Сложность:** $O(n^2)$.

Арифметические операции

TMatrix operator+(const TMatrix& mt)

- **Параметры:** mt — другая матрица того же размера.
- **Функционал:** поэлементное сложение двух матриц.
- **Сложность:** $O(n^2)$.

TMatrix operator-(const TMatrix& mt)

- **Параметры:** mt — другая матрица того же размера.
- **Функционал:** поэлементное вычитание двух матриц.

- **Сложность:** $O(n^2)$.

friend std::istream& operator>>(std::istream& in, TMatrix& mt)

- **Функционал:** считывает элементы матрицы из потока построчно.
- **Сложность:** $O(n^2)$.

friend std::ostream& operator<<(std::ostream& out, const TMatrix& mt)

- **Функционал:** выводит матрицу в поток в виде таблицы. Элементы ниже главной диагонали заменяются значениями по умолчанию (например, нулями).
- **Сложность:** $O(n^2)$.

Краткие комментарии к тестам

Для проверки корректности работы классов `TVector` и `TMatrix` были разработаны тесты с использованием фреймворка **Google Test**. Тесты охватывают базовые операции, граничные случаи и поведение при ошибках.

Тесты класса `TVector`

- **vector_length** – проверяет корректность создания вектора с заданной длиной и выброс исключения при отрицательном размере или превышении максимального.
- **vector_start_index** – проверяет корректность задания стартового индекса и выброс исключения при отрицательном значении.
- **copied_vector_is_equal** – проверяет работу конструктора копирования: созданный вектор равен исходному.
- **copied_vector_has_its_own_memory** – убеждается, что копия имеет собственную память и изменения не влияют на оригинал.
- **getter** – проверяет корректность методов `GetSize` и `GetStartIndex`.
- **vector_index_is_out_of_range** – проверяет выброс исключения при обращении к элементу вне диапазона.
- **equality_operator** – проверяет работу операторов сравнения `==` и `!=`.

- **assign_vector** – проверяет корректность оператора присваивания, включая самоприсваивание и присваивание векторов разного размера.
- **vector_and_digit** – проверяет арифметические операции с вектором и скаляром (+, -, *).
- **vectors_plus_minus** – проверяет поэлементное сложение и вычитание векторов одинакового размера, а также выброс исключения при разных размерах.
- **multiply_vectors** – проверяет корректность вычисления скалярного произведения и выброс исключения при несовпадении размеров.
- **output_operator** – проверяет корректность вывода вектора в поток.

Тесты класса TMatrix

- **matrix_length_and_creation** – проверяет корректность создания матрицы с заданным размером и выброс исключения при некорректных значениях.
- **copy_and_assigned_matrix** – проверяет работу конструктора копирования и оператора присваивания, включая независимость копии и корректность присваивания матриц разного размера.
- **equal_operator** – проверяет работу операторов сравнения == и !=.
- **plus_minus_operations** – проверяет корректность операций сложения и вычитания матриц одинакового размера, а также выброс исключения при разных размерах.
- **index_test** – проверяет доступ к элементам матрицы и выброс исключения при выходе за границы.
- **output_operator** – проверяет корректность вывода матрицы в поток, включая отображение нулей ниже главной диагонали.