Отчет по лабораторной работе №2

Предмет: «Алгоритмы и структуры данных»

Тема лабораторной работы: «Верхнетреугольные матрицы на шаблонах»

Выполнил:

Студент

Курс: 2

Группа: 3824Б1ФИ2

ФИО: Большаков Владислав Владимирович

2025 г.

Содержание:

1. Постановка задачи
2. Описание класса TVector, методов класса TVector
3. Описание класса TMatrix, методов класса TMatrix
4. Описание тестов Google Test

1. **Постановка задачи**

   Задача: разработка программных средств, поддерживающих эффективное хранение матриц специального вида (верхнетреугольных) и выполнение основных операций над ними: сложение, вычитание, копирование, сравнение. Разработка производится на языке программирования C++. Структурой данных для хранения верхнетреугольных матриц является класс TMatrix. Класс TMatrix является потомком класса TVector, который представляет собой структуру данных для хранения алгебраического вектора. Для проверки корректности работы реализованных классов производится написание автоматических тестов Google Test.

2. **Описание класса TVector, методов класса TVector.**

   Класс TVector предназначен для хранения алгебраических векторов и работы с ними. Реализованы такие операции векторной алгебры, как:

   - сложение/вычитание со скаляром, умножение на скаляр.
   - Сложение/вычитание векторов, скалярное произведение векторов.

   Данный класс является шаблонным (касается поля pVector).

   Описание полей (спецификатор доступа **protected**):

   - **Size** – максимальный размер объекта (вектора). Данное поле представлено типом int. Ограничение: максимальный размер определен константой типа int **MAX_VECTOR_SIZE** = 100 000 000. Также размер вектора не может быть меньше или равен 0 (проверяется в методах класса)
   - **StartIndex** – индекс первого элемента вектора. Подразумевается, что до стартового индекса все компоненты вектора являются нулями. Данное поле представлено типом int. Ограничения: не может быть отрицательным, наименьшее значение 0, не может быть больше или равен полю **Size** (проверяется в методах класса)
   - **pVector** – динамический массив типа **ValType** (**ValType** – шаблонный параметр. Позволяет массиву pVector хранить разные типы данных: int pVector, double pVector …). Память, выделяемая для массива, определяется по формуле: Size – StartIndex. Пример: Size = 10, StartIndex = 6. Тогда память выделиться под 4 компоненты вектора (индексация со значения StartIndex).

   Использование спецификатора protected позволяет классу TMatrix обращаться к данным полям (извне данные поля не доступны).

   Описание методов (у всех методов спецификатор **public**):

   Конструкторы и деструктор:

   **TVector(int s = 10, int si = 0) –** конструктор инициализатор.

   - Описание параметров и возвращаемых значений: принимает два целочисленных параметров типа int. Данные параметры имеют значения по умолчанию.
   - Описание реализуемого функционала: параметр s используется для инициализации поля Size, si – для инициализации поля StartIndex. На основе значений данных параметров выделяется память под определенное количество элементов (s-si). В случае некорректных значений s, si (см. описание полей класса TVector) выбрасываются исключения **invalid_argument**.

Также все элементы массива pVector инициализируются значениями по умолчанию.

- Оценка сложности: т.к. происходит не просто выделение памяти с помощью функции new, но ещё инициализации всех элементов значениями по умолчанию, то сложность оценивается как O(n), где n – количество элементов в массиве.

**TVector(const TVector& v) –** конструктор копирования.

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса.
- Описание реализуемого функционала: копирует состояние другого объекта в объект, для которого вызван данный конструктор (т.е. копируются значения полей: Size, StartIndex, pVector (копируются значения из ячеек массива, адреса памяти не копируется)).
- Оценка сложности: копирование значений элементов массива pVector происходит с помощью цикла for. Поэтому итоговая оценка O(n) (n = v.Size – v.StartIndex)

**~TVector() –** деструктор.

- Описание реализуемого функционала: освобождает память, выделенную для массива pVector. После освобождения присваивает указателю pVector значение nullptr.

**GetSize():**

- Описание параметров и возвращаемых значений: возвращает значение типа int.
- Описание реализуемого функционала: возвращает значение поля Size текущего объекта.

**GetStartIndex():**

- Описание параметров и возвращаемых значений: возвращает значение типа int.
- Описание реализуемого функционала: возвращает значение поля StartIndex текущего объекта.

**operator[](int pos)** – оператор индексирования.

- Описание параметров и возвращаемых значений: принимает значение типа int (является индексом интересующего нас элемента), возвращает ссылку на тип ValType (возвращение ссылки позволяет изменять состояние элемента по указанному индексу)
- Описание реализуемого функционала: проверяется корректность передаваемого индекса (не может быть меньше значения StartIndex, не может быть больше или равно значению поля Size). В случае передачи некорректного индекса бросается исключение **out_of_range**. После успешной проверки возвращается ссылка на соответствующий элемент внутреннего массива. Так как внутренний массив хранится с учетом сдвига, реальный индекс внутри массива рассчитывается как разница между передаваемой позицией и стартовым индексом.

**Перегруженный оператор = =**

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект данного класса, возвращает значение типа bool, является константным методом.
- Описание реализуемого функционала: сравниваются два объекта данного класса. Если объекта равны, то возвращается true, иначе false. Размеры объектов обязательно должны совпадать (иначе объекты не равны). При этом значения полей StartIndex могут отличаться.

Определяются минимальное и максимальное из значений StartIndex. Далее с помощью цикла for производится проход по элементам объекта, имеющего минимальный стартовый индекс. Данный проход осуществляется до максимального стартового индекса. При этом соответствующие ячейки должны иметь значение по умолчанию (в противном случае возвращается false). Далее с помощью цикла for происходит сравнение элементов двух объектов от максимального стартового индекса до конца массивов. Если есть не совпадающие элементы возвращается false. Пример:

v1 (Size = 6, StartIndex = 1)

| | 0 | 0 | 7 | 8 | 5 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

v2 (Size =6, StartIndex =3)

| | | | 7 | 8 | 5 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

Подразумевается, что до стартового индекса все компоненты вектора являются нулями. Поэтому данные векторы равны.

- Оценка сложности: в операторе задействованы два цикла for (первый проходит по вектору с минимальным стартовым индексом до максимального, второй сравнивает значения элементов вектора от максимального стартового индекса).
  В общей сумме сравнивается n элементов (количество элементов вектора с минимальным стартовым индексом). Итоговая оценка: O(n)

**Перегруженный оператор ! =**

- Описание параметров и возвращаемых значений: аналогично оператору = =.
- Описание реализуемого функционала: использует перегруженный оператор = = и инвертирует полученный результат (т.е. возвращается true, если объекты не равны; false, если объекты равны).
- Оценка сложности: аналогично оператору = =.

**Перегруженный оператор присваивания.**

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса, возвращает ссылку на текущий объект (*this). Возвращение ссылки на текущий объект позволяет делать запись «объект_1 = объект_2 = объект_3 = …» возможной.
- Описание реализуемого функционала: в первую очередь осуществляется проверка на самоприсваивание (если передан тот же самый объект, то сразу возвращается *this). Если объекты имеют равные размеры, то нет необходимости перевыделять память для исходного объекта. В этом случае происходит только копирование значений массива аргумента в массив текущего объекта. Если объекты имеют разные размеры (разные значения поля Size), то:
  1. Выделяется новая память подходящего размера (под количество элементов массива аргумента). Если произошла ошибка выделения памяти, то выбросится исключение, а память исходного объекта не будет освобождена.
  2. Происходит освобождение памяти текущего объекта, указателю pVector текущего объекта присваивается указатель на только что выделенную память.
  3. Указателю на новую выделенную память присваивается значение nullptr.
  4. Изменяются значения полей StartIndex, Size. Происходит копирование значений массива аргумента.

- Оценка сложности: из-за цикла for, с помощью которого осуществляется копирование значений массива аргумента, итоговая сложность O(n), n – количество элементов массива аргумента.

Скалярные операции

**Сложение вектора со скаляром:**

- Описание параметров и возвращаемых значений: принимает ссылку на константу типа ValType, возвращает новый объект TVector.
- Описание реализуемого функционала: создается новый объект класса TVector (размер и стартовый индекс такие же, как у текущего объекта). С помощью цикла for к каждой компоненте вектора прибавляется скаляр (если тип передаваемого аргумента не совпадает с типом вектора, то происходит неявное приведение типов)
- Оценка сложности: при создании объекта вызывается конструктор инициализатор (сложность O(n)). Цикл for проходит по каждой компоненте вектора (сложность также O(n)). Общая сложность O(2n). Итоговая оценка O(n).

**Вычитание из вектора скаляра:**

- Описание параметров и возвращаемых значений: аналогично операции сложения со скаляром.
- Описание реализуемого функционала: аналогично операции сложения со скаляром, только из каждой компоненты вычитается скаляр.
- Оценка сложности: аналогично операции сложения со скаляром.

**Умножение вектора на скаляр:**

- Описание параметров и возвращаемых значений: аналогично операции сложения со скаляром.
- Описание реализуемого функционала: аналогично операции сложения со скаляром, только каждая компонента умножается на скаляр.
- Оценка сложности: аналогично операции сложения со скаляром.

Векторные операции

**Сложение двух векторов:**

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса, возвращает новый объект, содержащий в себе результат сложения двух векторов.
- Описание реализуемого функционала:
  1. Проверяются размеры векторов (если размеры не равны бросается исключение **runtime_error**).
  2. Выбирается минимальный стартовый индекс, создается новый объект данного класса с минимальным стартовым индексом.
  3. Из объекта с минимальным стартовым индексом копируются значения компонент до максимального стартового индекса (в новый объект).
  4. От максимального стартового индекса идет сложение компонент двух векторов (полученные значения записываются в соответствующие компоненты нового вектора)

- Оценка сложности: создается новый объект (сложность O(n)). Как и в операторе = = происходит проход по всем компонентам вектора с минимальным стартовым индексом (сложность O(n)). Т.е. общая сложность O(2n). Итоговая оценка: O(n).

**Вычитание двух векторов.**

- Описание параметров и возвращаемых значений: аналогично сложению двух векторов (только новый объект содержит в себе результат разности двух векторов)
- Описание реализуемого функционала:
  1. Аналогично операции сложения выбирается минимальный стартовый индекс и создается новый объект.
  2. Если минимальный стартовый индекс – стартовый индекс текущего объекта (не аргумента), то аналогично операции сложения, только в качестве результата присваивается разность компонент.
  3. Если минимальный стартовый индекс – индекс аргумента, то сначала мы копируем значения текущего объекта (от стартового индекса текущего объекта до конца его массива). После производим разность компонент нового объекта (в несколько компонент которого уже записаны значения текущего объекта) с компонентами аргумента.

     Пример (для (3)):

     v1 (Size = 5, StartIndex = 2)

     | | | 2 | 3 | 4 |
     |---|---|---|---|---|
     | 0 | 1 | 2 | 3 | 4 |

     v2 (Size = 5, StartIndex = 1)

     | | 1 | 2 | 1 | 2 |
     |---|---|---|---|---|
     | 0 | 1 | 2 | 3 | 4 |

     Итоговый вектор: v3(Size = 5, StartIndex = 1)

     | | -1 | 0 | 2 | 2 |
     |---|---|---|---|---|
     | 0 | 1 | 2 | 3 | 4 |

- Оценка сложности: аналогично сложению двух векторов (во втором случае, когда минимальный стартовый индекс – индекс аргумента, происходит проход по всем элементам итогового вектора (совпадает с количеством элементов аргумента), также проход по элементам текущего вектора). Итоговая оценка: O(n)

**Скалярное произведение векторов:**

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса, возвращает переменную типа ValType.
- Описание реализуемого функционала:
  1. Проверяются размеры векторов (если размеры не равны, то бросается исключение **runtime_error**)
  2. Создается переменная типа ValType (принимает значение по умолчанию)
  3. Выбирается максимальный стартовый индекс (от максимального стартового индекса гарантированно существуют элементы в массивах обоих векторов).
  4. От максимального стартового индекса до Size происходит перемножение компонент векторов. Каждое произведение добавляется к промежуточной переменной (созданная переменная типа ValType), накапливающей итоговый результат

- Оценка сложности: цикл for проходит по минимальному из массивов двух векторов (по размеру). Итоговая оценка: O(n), n – количество элементов в минимальном массиве.

Ввод – вывод

**Перегруженный оператор ввода:**

- Описание параметров и возвращаемых значений: принимает два параметра. Первый – ссылка на объект типа **std::istream**, второй – ссылка на объект данного класса. Возвращает ссылку на объект istream (поэтому возможно применить несколько операторов подряд: **std::cin**>>объект_1>>объект_2).
- Описание реализуемого функционала:
  1. Создается новый объект данного класса, являющийся копией исходного вектора (на случай, если в несколько компонент удалось записать значения, а на каком-то шаге произошла ошибка ввода)
  2. Далее идет блок try…catch(). В блоке try с помощью цикла for в каждую компоненту из потока ввода записывается определенное значение. Если произошла ошибка, то с помощью функции **clear()** удаляется флаг ошибки (чтобы можно было произвести новый ввод), с помощью функции **ignore()** происходит удаление некорректных символов в потоке ввода (чтобы при новом вводе данные символы ни на что не повлияли), бросается исключение **invalid_argument**.
  3. Если возникла ошибка ввода, в блоке catch восстанавливается состояние исходного объекта с помощью резервной копии (вызывается оператор присваивания), после чего повторно бросается исключение, уведомляя пользователя о произошедшей ошибке
- Оценка сложности:

1. Создается копия исходного объекта (O(n))
2. Цикл for проходит по доступным компонентам вектора (O(n))
3. Если произошло исключение, то вызывается оператор присваивания (O(n))

Итоговая оценка: O(n)

**Перегруженный оператор вывода:**

- Описание параметров и возвращаемых значений: принимает два параметра. Первый – ссылка на объект типа **std::ostream**, второй – ссылка константный объект данного класса. Возвращает ссылку на объект ostream (поэтому возможно применить несколько операторов подряд: **std::cout**<<объект_1<<объект_2)
- Описание реализуемого функционала:

1. Создается переменная типа ValType, которая инициализируется значением по умолчанию.
2. С помощью цикла for происходит вывод данной переменной (количество выводов равно значению StartIndex)
3. С помощью второго цикла for через пробел выводятся компоненты вектора.

- Оценка сложности: Первый цикл выполняется ровно StartIndex раз, выводя начальные нулевые значения. Второй цикл проходит по всем действительным компонентам вектора, выполняя вывод каждой компоненты.
  Итоговая оценка: O(n), n – Size текущего объекта.

## 3. Описание класса TMatrix, методов класса TMatrix

Класс TMatrix предназначен для хранения верхнетреугольных матриц и работы с ними. Реализованы такие операции линейной алгебры, как:

- Сложение матриц
- Вычитание матриц

Данный класс также является шаблонным. Также данный класс является потомком класса TVector (public наследование). Каждый элемент массива pVector представлен отдельным объектом класса TVector<ValType>, где каждый вектор соответствует строке матрицы.

Ограничения: размер объекта данного класса (т.е. поле Size) не может быть меньше или равно 0, а также больше константы типа int MAX_MATRIX_SIZE = 10 000.

Описание полей: у данного класса отсутствуют собственные поля (только поля, унаследованные от класса TVector)

Описание методов (у всех методов спецификатор public):

**TMatrix(int s = 10) –** конструктор инициализатор

- Описание параметров и возвращаемых значений: принимает значение типа int. Данный параметр представляет собой размер матрицы (количество строк и столбцов). Имеет значение по умолчанию, равное 10.
- Описание реализуемого функционала: через списки инициализации вызывает конструктор инициализации класса TVector (передаваемые параметры: размер равный s, стартовый индекс равный 0). Имеется проверка корректности передаваемого размера. В случае некорректного значения (см. описание данного класса) выбрасывается исключение **invalid_argument**. После с помощью цикла for каждому элементу pVector присваивается объект класса TVector (при этом размер данных объектов один и тот же и равен переданному параметру s, а стартовый индекс увеличивается на 1 (от 0 до s-1)).
- Оценка сложности: в списках инициализации происходит выделение памяти под вектор векторов. Также внутри тела конструктора с помощью цикла for происходит перевыделение памяти для каждого элемента массива pVector (при этом память у каждого последующего элемента уменьшается на 1). Значит сложность O(s+s-1+s-2+…+1), что равняется O( ($s^2$+s)/2), т.е. O($s^2$). Значит итоговая сложность: O($s^2$)

**TMatrix(const TMatrix& mt) –** конструктор копирования

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса.
- Описание реализуемого функционала: в списках инициализации вызывается конструктор инициализации класса TVector (передаваемый размер равен размеру аргумента). В теле конструктора с помощью цикла for происходит копирование элементов массива pVector аргумента в массив pVector текущего объекта (для каждого элемента массива pVector текущего объекта вызывается оператор присваивания класса TVector)

- Оценка сложности: Цикл for проходит по все элементам pVector, для каждого элемента вызывается оператор присваивания класса TVector (сложность оператора O(n)). Тогда общая сложность цикла $O(n^2)$. Итоговая сложность: $O(n^2)$

**TMatrix(const TVector<TVector<ValType>>& mt) –** конструктор преобразования типа.

- Описание параметров и возвращаемых значений: передается ссылка на константный объект класса TVector (на вектор векторов).
- Описание реализуемого функционала: данный конструктор предназначен для преобразования объекта класса TVector<TVector<ValType>> (вектора векторов) в объект класса TMatrix.
  1. Создается не константная копия вектора векторов (иначе использовать перегруженный оператор индексации класса TVector не получится)
  2. Присутствуют проверки на корректность размера данного вектора (не больше, чем MAX_MATRIX_SIZE, иначе бросается исключение **invalid_argument**), на корректность стартового индекса данного объекта (должен быть равен 0, иначе бросается исключение **invalid_argument**), каждый внутренний вектор должен иметь свой стартовый индекс, увеличивающийся от вектора к вектору (это свойство требуется для формирования верхнетреугольной структуры; иначе бросается исключение **runtime_error**).
  3. Выделяется память под вектор векторов. Указателю pVector присваивает указатель на только что выделенную память (а после указателю на выделенную память присваивается nullptr (не указателю pVector)). Копируется значение поля Size.
  4. С помощью цикла for каждый элемент pVector копирует состояние соответствующего элемента вектора векторов (используется оператор присваивания класса TVector)
- Оценка сложности: вызов оператора присваивания в цикле for ($O(n^2)$)
  Итоговая оценка: $O(n^2)$

**Перегруженный оператор = =**

- Описание параметров и возвращаемых значений: принимает ссылку на константу данного класса, возвращает значение типа bool, является константным методом
- Описание реализуемого функционала: сравнивает два объекта класса TMatrix. Если не совпадают размеры (поле Size) или есть отличия в какой-либо из строк (строка – объект класса TVector), то возвращается false. Иначе true.
- Оценка сложности: в цикле for проверяется совпадение строк. Для каждой строки (вектора) вызывается перегруженный оператор = = класса TVector. Его сложность O(n), где n – количество элементов вектора, содержащихся в массиве pVector. Пусть размер матрицы равен n. Количество элементов в каждом векторе уменьшается на единицу (содержащихся в массивах pVector). Значит сложность $O((n^2+n)/2)$. Итоговая сложность $O(n^2)$.

**Перегруженный оператор !=**

- Описание параметров и возвращаемых значений: аналогично оператору = = данного класса
- Описание реализуемого функционала: использует оператор = = данного класса и инвертирует его результат (если матрицы равна, то возвращает false, иначе true)

- Оценка сложности: $O(n^2)$

**Перегруженный оператор присваивания:**

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса, возвращает ссылку на текущий объект данного класса (благодаря этому возможна запись: «объект_1 = объект_2 = объект_3 = …»)
- Описание реализуемого функционала: аналогично оператору присваивания в классе TVector. Отличия: выделяется память под вектор векторов, значение StartIndex не копируется (т.к. его нельзя установить в конструкторе инициализации), в цикле for используется оператор присваивания класса TVector (проходится по всем строка матрицы).
- Оценка сложности: пусть размер передаваемой матрицы равен n. Вызов оператора присваивания класса TVector в цикле for: $O((n^2+n)/2)$. Итоговая оценка: $O(n^2)$.

**Сложение матриц (перегруженный оператор +):**

- Описание параметров и возвращаемых значений: принимает ссылку на константный объект данного класса, возвращает новый объект данного класса, содержащий результат сложения двух матриц.
- Описание реализуемого функционала:
  1. Если размеры матриц не совпадают, то бросается исключение **runtime_error**.
  2. Создается новый объект класса TMatrix.
  3. В цикле for каждой строке нового объекта присваивается результат сложения двух векторов (соответствующий вектор текущего объекта и соответствующий вектор аргумента).
- Оценка сложности: пусть размер матрицы равен n
  1. В цикле for сначала вызывается «оператор +» класса TVector
  2. После полученный результат записывается в соответствующую строку (вектор) нового объекта TMatrix. Для этого используется оператор = класса TVector. Сложность данных операций в цикле for оценим $O(n^2)$
  3. Также новый объект класса TMatrix возвращается в качестве результата работы оператора +. Для этого вызывается конструктор копирования класса TMatrix. Его сложность $O(n^2)$

  Итоговая оценка $O(n^2)$

**Вычитание двух матриц (перегруженный оператор -):**

- Описание параметров и возвращаемых значений: аналогично «оператору +» данного класса.
- Описание реализуемого функционала: аналогично «оператору +» данного класса. Только в цикле for вызывается «оператор -» класса TVector.
- Оценка сложности: аналогично «оператору +» данного класса (т.е. $O(n^2)$).

**Перегруженный оператор ввода:**

- Описание параметров и возвращаемых значений: аналогично оператору ввода класса TVector.
- Описание реализуемого функционала:

1. Создается новый объект данного класса, являющийся копией текущего объекта (в тех же целях, что и в классе TVector)
2. Цикл for проходит по всем строкам текущего объекта. В данном цикле присутствует блок try…catch. В блоке try используется оператор ввода класса TVector для соответствующей строки матрицы (вектора). Если ввод не удачный, то (из-за бросаемого исключения в операторе ввода класса TVector) происходит переход в блок catch. В данном блоке текущему объекту присваивается его копия. После бросается исключение **invalid_argument.**

- Оценка сложности: пусть размер матрицы равен n
  1. Создание копии исходной матрицы с помощью конструктора копирования: $O(n^2)$
  2. Вызов оператора ввода класса TVector в цикле for: $O(n^2)$
  3. В случае ошибки ввода происходит присваивание текущему объекту копии с помощью оператора присваивания данного класса: $O(n^2)$

  Итоговая оценка $O(n^2)$

**Перегруженный оператор вывода:**

- Описание параметров и возвращаемых значений: аналогично оператору вывода класса TVector.
- Описание реализуемого функционала: в цикле for происходит вывод каждой строки матрицы с помощью оператора вывода класса TVector. Каждый вывод строки оканчивается символом «\n»
- Оценка сложности: пусть размер матрицы равен n
  1. Сложность оператора вывода класса TVector: O(n)
  2. Из-за использования оператора вывода класса TVector в цикле for, итоговая сложность оценивается как $O(n^2)$.

## 4. Описание тестов Google test

**Тесты для класса TVector.**

В группе тестов **TVector_test_0** проверяется корректность работы конструктора инициализации, оператора индексации, конструктора копирования.

Конструктор инициализации: установка значений по умолчанию (поля Size, StartIndex, а также инициализация элементов массива pVector значениями по умолчанию), установка собственных значений (поля Size, StartIndex), выброс исключения при попытке установки некорректных значений, создание объекта с максимально допустимой длиной (Size = MAX_VECTOR_SIZE).

Оператор индексации: установка значений для конкретной компоненты вектора (в зависимости от стартового индекса), выброс исключения при попытке обращения к элементу по некорректному индексу, последовательное применение данного оператора к одной и той же компоненте (изменение значения данной компоненты).

Конструктор копирования: успешность копирования объекта, копирования объекта большого размера, независимость копии и оригинального объекта (изменение копии не приводит к изменению оригинального объекта).

В группе тестов **TVector_test_1** проверяется корректность работы оператора = =, оператора !=, оператора присваивания.

Оператор = = : сравнение объектов с одинаковыми значениями полей (включая совпадение элементов массива pVector); сравнение объектов с разными значениями полей Size, StartIndex, с отличиями в компонентах векторов.

Оператор != : сравнение объектов с отличиями в значениях полей, сравнение объектов с одинаковыми значениями полей.

Оператор присваивания: корректность работы данного оператора; рассмотрены случаи самоприсваивания, применения нескольких операторов подряд.

В группе тестов **TVector_test_2** проверяется корректность применения скалярных операций (сложение вектора со скаляром, вычитание из вектора скаляра, умножение вектора на скаляр).

Сложение вектора со скаляром: корректность работы данной операции (скаляры разных типов).

Вычитание из вектора скаляра: аналогично операции сложения со скаляром.

Умножение вектора на скаляр: корректность работы данной операции, не совпадение типов вектора и скаляра.

В группе тестов **TVector_test3** проверяется корректность применения векторных операций (сложение двух векторов, разность двух векторов, скалярное произведение двух векторов).

Сложение двух векторов: корректность работы данной операции при совпадении у векторов значений полей Size, StartIndex; корректность работы для векторов с разными стартовыми индексами, выброс исключения при попытке применения данной операции к векторам разных размеров.

Разность двух векторов: аналогично операции сложения двух векторов.

Скалярное произведение двух векторов: корректность работы данной операции при совпадении у векторов значений полей Size, StartIndex; корректность работы применительно к векторам с разными стартовыми индексами, запись результата операции в переменную другого типа (отличного от типа векторов), выброс исключения в ситуации аналогичной сложению двух векторов.

В группе тестов **TVector_test4** проверяется корректность работы операторов ввода и вывода.

Оператор ввода: производится ввод значений компонент вектора (в зависимости от стартового индекса соответствующее значение записывается в соответствующую компоненту). Рассмотрены следующие случаи: ввод значений через пробел, запятую; попытка ввода значения, имеющего тип, отличный от типа вектора (соответственно выброс исключения); сохранение данных, содержащихся в векторе, при выбросе исключения; попытка ввода большего/меньшего количества значений чем размер массива вектора; применение нескольких операторов подряд.

Оператор вывода: корректность работы данного оператора при разных значениях стартового индекса вектора, применение нескольких операторов подряд.

**Тесты для класса TMatrix.**

В группе тестов **TMatrix_test1** проверяется корректность конструктора инициализации, конструктора копирования, конструктора преобразования типа.

Конструктор инициализации: установка значений по умолчанию, установка собственных значений (поле Size, при этом элементы матрицы имеют один и тот же размер, а стартовые индексы увеличиваются на 1 от элемента к элементу), выброс исключения при попытке создания объекта с некорректным размером, создание объекта с максимально возможным размером (Size = MAX_MATRIX_SIZE).

Конструктор копирования: успешность копирования объекта, независимость копии и оригинального объекта (изменение копии не приводит к изменению оригинального объекта).

Конструктор преобразования типа: работа данного конструктора при корректных данных; попытка преобразования вектора векторов, элементы которого имеют одинаковые стартовые индексы (выброс исключения); попытка преобразования вектора векторов, размер которого превосходит максимально допустимый для матрицы размер (выброс исключения); попытка преобразования вектора векторов, стартовый индекс которого не равен 0 (выброс исключения).

В группе тестов **TMatrix_test2** проверяется корректность работы оператора = =, оператора !=, оператора присваивания, операций сложения и вычитания матриц.

Оператор = = : сравнение одинаковых матриц (по значению поля Size и по элементам); сравнение матриц, имеющих разные значения поля Size, разные элементы.

Оператор != : аналогично оператору = = .

Оператор присваивания: корректность работы данного оператора; изменение объекта, который вызвал данный оператор, не ведет к изменению второго объекта (с данными которого работал оператор присваивания); случай самоприсваивания; применение нескольких операторов подряд.

Сложение матриц: корректность работы данной операции; попытка применения данной операции к матрицам, имеющим разные размеры.

Разность двух матриц: корректность работы данной операции.

В группе тестов **TMatrix_test3** проверяется корректность работы операторов ввода и вывода.

Оператор ввода: ввод значений матрицы через пробел, с использованием символа «\n»; ввод большего количества значений, чем может содержать верхнетреугольная матрица данного размера; попытка ввод некорректных значений (имеющих тип, отличный от типа матрицы; при этом значения, содержащиеся в матрице, сохранились); применение нескольких операторов подряд.

Оператор вывода: корректность работы данного оператора; применение нескольких операторов подряд.