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

Отчёт по лабораторной работе №2
дисциплины «Алгоритмы и структуры данных»

Выполнил:

Студент группы 3824Б1ФИ2

Старостин Д.Д.

Нижний Новгород – 2025 г.

1. Постановка задачи

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

2. Описание программной реализации

2.1.1. Описание класса TVector

Класс TVector реализует шаблонный математический вектор с произвольным выбором индекса первого элемента. Содержит поля:

- Size – размер вектора;
- StartIndex – индекс первого элемента вектора;
- pVector – указатель на память для хранения элементов вектора.

Методы класса обеспечивают доступ к полям Size и StartIndex и к элементам вектора, сравнение объектов класса, математические операции над вектором, а также ввод и вывод данных.

2.1.2Описание методов класса и функций

1) TVector(int s = 10, int si = 0) – конструктор с параметрами s(размер) и si(начальный индекс) со значениями по умолчанию 10 и 0 соответственно.
   - Параметры и возвращаемые значения: int s – задаёт размер поля, значение по умолчанию – 10; int si – задаёт индекс первого элемента вектора, значение по умолчанию – 0. Проверяет корректность параметров.
   - Функционал: присваивает полю Size значение s, полю StartIndex – si. Выделяет динамическую память под массив из s - si элементов и присваивает полю pVector указатель на начало этого массива. Элементы pVector по умолчанию инициализируются конструктором по умолчанию типа ValType.
   - Сложность: O(s - si), так как выделяется память для массива из s - si элементов.

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

- Параметры и возвращаемые значения: v – константная ссылка на копируемый объект.
- Функционал: копирует объект v.
- Сложность: O(v.Size - v.StartIndex). Выделение памяти, копирование элементов v.

3) ~TVector() – деструктор класса.
- Функционал: освобождает память поля pVector.

4) int GetSize()
- Параметры и возвращаемые значения: возвращает значение поля Size.
- Функционал: предоставляет доступ к значению поля size.

5) int GetStartIndex ()
- Параметры и возвращаемые значения: возвращает значение поля StartIndex.
- Функционал: предоставляет доступ к значению поля StartIndex.

6) ValType& operator[](int pos) – доступ к элементу.
- Параметры и возвращаемые значения: int pos – позиция возвращаемого элемента в векторе. Возвращает ссылку на элемент вектора с индексом pos - StartIndex. Допустимые значения параметра: StartIndex <= pos < Size.
- Функционал: предоставляет доступ к чтению и изменению элемента на позиции pos в векторе. Проверяет корректность параметра.

7) bool operator==(const TVector &v) const – оператор проверки равенства векторов.
    - Параметры и возвращаемые значения: const TVector &v – константная ссылка на объект, с которым происходит сравнение. Возвращает false, если векторы не равны или имеют разные размеры, иначе true.
    - Функционал: производит проверку равенства векторов. Векторы считаются равными, если у них одинаковые размеры и элементы.
    - Сложность: O(Size - min(StartIndex)), так как поэлементно сравнивает векторы. O(1) – в лучшем случае(векторы имеют разные размеры).

8) bool operator!=(const TVector &v) const – оператор проверки отсутствия равенства векторов.

- Параметры и возвращаемые значения: const TVector &v – константная ссылка на объект, с которым происходит сравнение. Возвращает true, если векторы не равны или имеют разные размеры, иначеfalse.
- Функционал: производит проверку отсутствия равенства между векторами. Векторы считаются равными, если у них одинаковые размеры и элементы.
- Сложность: O(Size - min(StartIndex)). O(1) – в лучшем случае. Аналогично сложности оператора ==, так как реализован через него.

9) TVector& operator=(const TVector &v) – оператор присваивания векторов.

- Параметры и возвращаемые значения: const TVector &v – константная ссылка на объект-источник данных. Возвращает ссылку на *this после присваивания.
- Функционал: обеспечивает копирование данных из объекта v в текущий объект. Осуществляет проверку на самоприсваивание.
- Сложность: O(v.Size – v.StartIndex), так как происходит поэлементное копирование, а также выделение памяти размера v.Size – v.StartIndex, если размеры не равны.

10) TVector operator+(const ValType &val) – прибавление скаляра.

- Параметры и возвращаемые значения: const ValType &val – константная ссылка на прибавляемый скаляр. Возвращает модифицированный объект.
- Функционал: прибавляет скаляр val к вектору начиная со StartIndex.
- Сложность: O(Size – StartIndex). Конструктор копирования с параметром *this, цикл из Size – StartIndex итераций в которых происходит сложение.

11) TVector operator-(const ValType &val) – вычитание скаляра.

- Параметры и возвращаемые значения: const ValType &val – константная ссылка на вычитаемый скаляр. Возвращает модифицированный объект.
- Функционал: вычитает скаляр val из вектора начиная со StartIndex.

- Сложность: O(Size – StartIndex). Конструктор копирования с параметром *this, цикл из Size – StartIndex итераций, в которых происходит вычитание.

12) TVector operator*(const ValType &val) – умножение на скаляр.
- Параметры и возвращаемые значения: const ValType &val – константная ссылка на множитель. Возвращает модифицированный объект.
- Функционал: вектор на скаляр val начиная со StartIndex.
- Сложность: O(Size – StartIndex). Конструктор копирования с параметром *this, цикл из Size – StartIndex итераций, в которых происходит умножение.

13) TVector operator+(const TVector &v) – прибавление вектора.
- Параметры и возвращаемые значения: const TVector &v – константная ссылка на правый аргумент. Возвращает модифицированный объект.
- Функционал: складывает векторы одинакового размера. Если объекты имеют разные StartIndex, модифицированный объект имеет наименьший из начальных индексов двух объектов.
- Сложность: O(Size - min(StartIndex)). Конструктор копирования, цикл из Size - max(StartIndex) итераций.

14) TVector operator-(const TVector &v) – вычитание вектора.
- Параметры и возвращаемые значения: const TVector &v – константная ссылка на правый аргумент. Возвращает модифицированный объект.
- Функционал: вычитает v из *this, если они имеют одинаковый размер. Если объекты имеют разные StartIndex, модифицированный объект имеет наименьший из начальных индексов двух объектов. Сложность: O(Size - min(StartIndex)). Конструктор копирования, цикл из Size - max(StartIndex) итераций.

15) ValType operator*(const TVector &v) – скалярное произведение векторов.
- Параметры и возвращаемые значения: const TVector &v – константная ссылка на правый аргумент. Возвращает скалярное произведение *this и v.

- Функционал: вычисляет и возвращает скалярное произведение векторов одинакового размера.
  Сложность: O(Size - max(StartIndex)). Цикл из Size - max(StartIndex) итераций.

16) friend std::istream& operator>>(std::istream& in, TVector& v) – ввод.
- Параметры и возвращаемые значения: std::istream &in – ссылка на поток ввода. TVector &v – ссылка на объект, в который происходит ввод. Возвращает ссылку на объект in.
- Функционал: осуществляет ввод объекта класса TVector.
- Сложность: O(v.Size - v.StartIndex).

17) friend std::ostream& operator<<(std::ostream& out, const TVector& v) – вывод.
- Параметры и возвращаемые значения: std::ostream &out – ссылка на поток вывода. const TVector &v – ссылка на выводимый объект. Возвращает ссылку на объект out.
- Функционал: осуществляет вывод объекта класса TVector.
- Сложность: O(v.Size). Выводит 2*v.Size символов.

2.2.1. Описание класса TMatrix

Класс TMatrix реализует шаблонный класс для работы с верхнетреугольными матрицами произвольного размера с шаблонным параметром ValType. Публично наследуется от класса TVector<TVector<ValType>>. Каждая строка матрицы представлена вектором типа TVector<ValType>. Для i-й строки начальный индекс равен i, где i принадлежит множеству от 0 до размера матрицы - 1, что обеспечивает верхнетреугольную структуру.

Методы класса обеспечивают сравнение объектов класса, их сложение, вычитание и присваивание, а также ввод и вывод данных. Унаследованные методы позволяют получить доступ к элементам матрицы и к её размеру.

2.2.2. Описание методов класса и функций

1) TMatrix(int s = 10) – конструктор с параметром размера.

- Параметры и возвращаемые значения: int s – задаёт размер матрицы, значение по умолчанию – 10.
- Функционал: присваивает полю Size значение s, полю StartIndex – 0. Выделяет динамическую память под массив из s элементов и присваивает полю pVector указатель на начало этого массива. Инициализирует строки верхнетреугольной матрицы объектами класса TVector<ValType>. i-я строка имеет StartIndex = i, где i = 0, …, s - 1.
- Сложность: $O(s^2)$. В цикле вызывается конструктор класса TVector<ValType>(s, i). То есть если при вызове каждого конструктора выполняется минимум s - i операций, то: $s – 0 + s – 1 + … + 1 = s(s + 1)/2 = O(s^2)$.

2) TMatrix(const TMatrix &mt) – конструктор копирования.

- Параметры и возвращаемые значения: const TMatrix &mt – константная ссылка на копируемый объект.
- Функционал: копирует объект mt.
- Сложность: $O(mt.Size^2)$. Выделение памяти, копирование элементов каждой строки mt. При копировании строк происходит вызов оператора присваивания. Вычисления аналогичны приведённым в пункте 1)

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

- Параметры и возвращаемые значения: const TVector<TVector<ValType>> &mt – константная ссылка на преобразовываемый объект.
- Функционал: преобразует объект mt к типу TMatrix.
- Сложность: $O(mt.Size^2)$. Получение неконстантной копии mt путём вызова оператора копирования класса TVector<TVector<ValType>> $O(mt.Size^2)$. Выделение памяти, копирование элементов каждой строки mt. При преобразовании элементов mt к строкам mt происходит вызов оператора присваивания. Вычисления аналогичны приведённым в пункте 1)

4) bool operator==(const TMatrix &mt) const –проверка равенства матриц.

- Параметры и возвращаемые значения: const TMatrix &mt – константная ссылка на объект, с которым происходит сравнение.

Возвращает false, если матрицы не равны или имеют разные размеры, иначе true.

- Функционал: производит проверку равенства матриц.
- Сложность: O($Size^2$), так как поэлементно сравнивает векторы векторов. O(1) – в лучшем случае(матрицы имеют разные размеры).

5) bool operator!=(const TMatrix &mt) const – проверка отсутствия равенства матриц.
   - Параметры и возвращаемые значения: const TMatrix &mt – константная ссылка на объект, с которым происходит сравнение. Возвращает true, если матрицы не равны или имеют разные размеры, иначе false.
   - Функционал: производит проверку отсутствия равенства матриц.
   - Сложность: O($Size^2$), так как поэлементно сравнивает векторы векторов. O(1) – в лучшем случае(матрицы имеют разные размеры).

6) TMatrix& operator= (const TMatrix &mt) – оператор присваивания.
   - Параметры и возвращаемые значения: const TMatrix &mt – константная ссылка на объект-источник данных. Возвращает ссылку на *this после присваивания.
   - Функционал: обеспечивает копирование данных из объекта mt в текущий объект. Осуществляет проверку на самоприсваивание.
   - Сложность: O(mt.$Size^2$), так как происходит поэлементное копирование.

7) TMatrix operator+ (const TMatrix &mt) – оператор сложения матриц.
   - Параметры и возвращаемые значения: const TVector &mt – константная ссылка на правый аргумент. Возвращает модифицированный объект.
   - Функционал: складывает матрицы одинакового размера.
   - Сложность: O($Size^2$). Конструктор копирования, Size операций сложения векторов.

8) TMatrix operator- (const TMatrix &mt) – оператор вычитания матриц.
   - Параметры и возвращаемые значения: const TVector &mt – константная ссылка на правый аргумент. Возвращает модифицированный объект.

- Функционал: вычитает матрицу mt из матрицы *this. Матрицы должны быть одинакового размера.
- Сложность: O($Size^2$). Конструктор копирования, Size операций вычитаний векторов.

9) friend std::istream& operator>>(std::istream& in, TMatrix& mt) – ввод матрицы.
   - Параметры и возвращаемые значения: std::istream &in – ссылка на поток ввода. TVector &mt – ссылка на объект, в который происходит ввод. Возвращает ссылку на объект in.
   - Функционал: осуществляет ввод объекта класса TMatrix.
   - Сложность: O(mt.$Size^2$). При вводе mt[i] выполняется минимум mt[i].Size операций. mt[0].size = mt.size. mt.size + mt.size – 1 + … + 1 = mt.Size(mt.Size + 1)/2 = O(mt.$Size^2$).

10) friend std::ostream& operator<<(std::ostream& out, const TMatrix& mt) – вывод.
    - Параметры и возвращаемые значения: std::ostream &out – ссылка на поток вывода. const TMatrix & mt – ссылка на выводимый объект. Возвращает ссылку на объект out.
    - Функционал: осуществляет вывод объекта класса TMatrix.
    - Сложность: O(mt.$Size^2$). Выводит mt.$Size^2$ символов.

3. Тесты

3.1. Тесты TVector

1) Блок TVector_Constructor:
   1.1) Size_Constructor: проверка корректности работы конструктора по умолчанию и конструктора от параметров s, si.
   1.2) Copy_Constructor: проверка корректной работы конструктора копирования, а также того, что изменение копии не изменяет оригинал.

2) Блок TVector_Get:

Get_Size_And_StartIndex: Проверка корректности работы методов Get_Size и Get_ StartIndex.

3) Блок TVector_Operators:
   3.1) Access: проверка корректности работы опреатора [].
   3.2) Equal: осуществляется проверка работы опреатора == равенства для идентичных объектов, объектов, содержащих одни и те же элементы с одинаковым StartIndex, объектов, содержащих одни и те же элементы с разными StartIndex, и отсутствия равенства объектов с разными размерами, разных объектов с разными элементами.
   3.3) Not_Equal: проверка корректности работы оператора != в вышеописанных случаях.
   3.4) Assignment: проверка корректной работы оператора = для разных объектов.
   3.5) Self_Assignment: проверка корректной работы оператора = при самоприсваивании.

4) Блок TVector_Scalar_Operations:
   4.1) Addition: проверка корректной работы операции прибавления скаляра.
   4.2) Subtraction: проверка корректной работы операции вычитания скаляра.
   4.3) Multiplication: проверка корректной работы операции умножения на скаляр.

5) Блок TVector_Vector_Operations:
   5.1) Addition: проверка сложения векторов с разными начальными индексами.
   5.2) Subtraction_Equal_Sizes: проверка вычитания векторов с одинаковыми начальными индексами.
   5.3) Subtraction_Differernt_Sizes: проверка вычитания векторов с разными начальными индексами.
   5.4) Product_Equal_Sizes: проверка операции скалярного умножения с равными начальными индексами.
   5.5) Product_Differernt_Sizes: проверка операции скалярного умножения с разными начальными индексами.

6) Блок TVector_Exceptions:
   6.1) Constructor: проверка возникновения ошибок при попытке создать вектор с некорректным размером или начальным индексом.
   6.2) Access: проверка возникновения ошибок при попытке доступа к элементу вне заданного отрезка.
   6.3) Vector_Operations: проверка возникновения ошибок при попытке сложения/вычитания/нахождения скалярного произведения векторов с разными размерами.

3.2. Тесты TMatrix

1) Блок TMatrix_Constructor:
   1.1) Size_Constructor: проверка корректности работы конструктора по умолчанию и конструктора от параметра s.
   1.2) Copy_Constructor: проверка корректной работы конструктора копирования, а также того, что изменение копии не изменяет оригинал.
   1.3) Converting_Constructor: проверка конструктора преобразования типа.
2) Блок TMatrix_Operators:
   2.1) Equal: проверка оператора ==. Осуществляется проверка равенства идентичных объектов, неравенства разных объектов, равного размера, содержащих одинаковые элементы и неравенства разных объектов с разными элементами.
   2.2) Not_Equal: проверка работы оператора != в случаях, описанных в пункте 2.1).
   2.3) Assignment: проверка корректной работы оператора = для разных объектов.
   2.4) Self_Assignment: проверка корректной работы оператора = при самоприсваивании.
   2.5) Addition: проверка корректной работы сложения матриц.
   2.6) Subtraction: проверка корректной работы вычитания матриц.
3) Блок TMatrix_Exceptions:
   3.1) Size_Constructor: проверка возникновения ошибок при попытке создать матрицу с некорректным размером.
   3.2) Converting_Constructor: проверка возникновения ошибки в конструкторе преобразования при невозможности преобразования вектора к верхнетреугольной матрице, то есть

при недопустимом начальном индексе или нарушении верхнетреугольной структуры строк.

3.3) Operators: проверка возникновения ошибок при попытке сложения/вычитания матриц с разными размерами.