ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ НИЖЕГОРОДСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО»

Отчет по лабораторной работе №1 по дисциплине «Алгоритмы и структуры данных»

Выполнил:
Студент Ненев А.Е.
Группа 3824Б1ФИ2
Нижний Новгород

2025г

1.Постановка задачи

Разработать структуру данных верхнетреугольную матрицу и методы работы с ней с помощью реализации класса вектор, а также проверка их корректной работы с помощью функционала Google Tests.

2. Класс TVector<ValType>

Описание класса

Шаблонный класс для представления вектора с динамическим выделением памяти. Поддерживает концепцию "стартового индекса" (StartIndex), позволяющую экономить память при работе с разреженными векторами. Элементы до StartIndex считаются нулевыми и не хранятся в памяти. Сам вектор хранит Size – StartIndex элементов.

Поля класса

Int Size Размер вектора (общее количество элементов)

Int StartIndex Индекс первого ненулевого элемента

ValType* pVector Указатель на динамический массив элементов выбранного типа данных

Методы класса 1. Конструктор TVector(int s = 10, int si = 0)

Параметры:

s - размер вектора (по умолчанию 10)

si - индекс первого элемента (по умолчанию 0)

Функционал: Создает вектор заданного размера с указанным стартовым индексом. Выделяет динамическую память под массив элементов. Проверяет корректность входных данных.

2. Конструктор копирования TVector(const TVector& v)

Параметры:

v - вектор-источник для копирования

Функционал: Создает глубокую копию вектора. Копирует Size, StartIndex и все элементы начиная с StartIndex. Сложность: O(n), где n = Size – StartIndex

3. Деструктор ~TVector()

Функционал: Освобождает динамически выделенную память и обнуляет указатель.

4. Метод GetSize() const

Возвращаемое значение: int - размер вектора

Функционал: Возвращает размер вектора

5. Метод GetStartIndex() const

Возвращаемое значение: int - индекс первого элемента

Функционал: Возвращает индекс первого ненулевого элемента

6. Оператор индексации operator[](int pos)

Параметры:

pos - индекс элемента

Возвращаемое значение: ValType& - ссылка на элемент массива

Функционал: Предоставляет доступ к элементу вектора по индексу. Причем, так как в памяти хранятся лишь элементы, начиная с StartIndex то для корректного исполнения возвращает pVector[pos – StartIndex].
Проверяет есть ли в данной позиции инициализированный элемент.

7. Оператор сравнения operator==(const TVector& v) const

Параметры:

v - вектор для сравнения

Возвращаемое значение: bool - результат сравнения

Функционал: Проверяет равенство двух векторов. Векторы равны, если совпадают Size, StartIndex и все элементы с индексами от StartIndex до Size.

Сложность: O(n), где n = Size - StartIndex

8. Оператор неравенства operator!=(const TVector& v) const

Параметры:

v - вектор для сравнения

Возвращаемое значение: bool - результат сравнения

Функционал: Возвращает отрицание результата оператора ==

Сложность: O(n), где n = Size - StartIndex (вызывает operator==)

9. Оператор присваивания operator=(const TVector& v)

Параметры:

v - вектор-источник

Возвращаемое значение: TVector& - ссылка на текущий объект

Функционал: Выполняет копирование вектора. Проверяет само-присваивание. Удаляет старую память, выделяет новую и копирует все параметры и элементы.

Сложность: O(n), где n = Size - StartIndex

10. Оператор сложения со скаляром operator+(const ValType& val)

Параметры:

val - скалярное значение для сложения

Возвращаемое значение: TVector - новый вектор-результат

Функционал: Прибавляет скаляр ко всем элементам вектора, включая нулевые (элементы до StartIndex). Результирующий вектор имеет StartIndex = 0, так как все элементы становятся ненулевыми.

Сложность: O(n), где n = Size

11. Оператор вычитания скаляра operator-(const ValType& val)

Параметры:

val - скалярное значение для вычитания

Возвращаемое значение: TVector - новый вектор-результат

Функционал: Вычитает скаляр из всех элементов вектора. Элементы до StartIndex (нули) превращаются в -val. Результирующий вектор имеет StartIndex = 0.

Сложность: O(n), где n = Size

12. Оператор умножения на скаляр operator*(const ValType& val)

Параметры:

val - скалярное значение для умножения

Возвращаемое значение: TVector - новый вектор-результат

Функционал: Умножает все ненулевые элементы вектора на скаляр. Нулевые элементы (до StartIndex) остаются нулями. Результирующий вектор сохраняет тот же StartIndex.

Сложность: O(n), где n = Size - StartIndex

13. Оператор сложения векторов operator+(const TVector& v)

Параметры:

v - вектор для сложения

Возвращаемое значение: TVector - новый вектор-результат

Функционал: Складывает два вектора поэлементно, причем если у векторов разные StartIndex, то начинаем с минимального и складываем его значения с нулями. Векторы должны иметь одинаковый Size. Результирующий вектор имеет StartIndex = min(StartIndex_1, StartIndex_2).

Сложность: O(n), где n = Size

14. Оператор вычитания векторов operator-(const TVector& v)

Параметры:

v - вектор для вычитания

Возвращаемое значение: TVector - новый вектор-результат

Функционал: Вычитает вектор v из текущего вектора поэлементно. Векторы должны иметь одинаковый Size. Результирующий StartIndex = min(StartIndex_1, StartIndex_2).

Сложность: O(n), где n = Size

15. Скалярное произведение operator*(const TVector& v)

Параметры:

v - вектор для скалярного произведения

Возвращаемое значение: ValType - результат скалярного произведения

Функционал: Вычисляет скалярное произведение двух векторов. Векторы должны иметь одинаковый Size. Умножение начинается с max(StartIndex_1,

StartIndex_2), так как до этого индекса хотя бы один из множителей равен нулю.

Сложность: O(Size - max(StartIndex_1, StartIndex_2))

16. Оператор ввода operator>>

Параметры:

in - входной поток

v -вектор для заполнения

Возвращаемое значение: std::istream& - ссылка на поток

Функционал: Читает элементы вектора из потока, начиная с индекса StartIndex до Size. Элементы до StartIndex полагаются нулями.

Сложность: O(n), где n = Size - StartIndex

17. Оператор вывода operator<<

Параметры:

out - выходной поток

v - вектор для вывода

Возвращаемое значение: std::ostream& - ссылка на поток

Функционал: Выводит вектор в поток. Сначала выводит нули для элементов до StartIndex, затем реальные значения элементов от StartIndex до Size.

Сложность: O(n), где n = Size

3. Класс TMatrix<ValType>

Описание класса

Шаблонный класс для представления верхнетреугольной матрицы. Наследуется от TVector<TVector<ValType>>. В верхнетреугольной матрице элементы ниже главной диагонали равны нулю и не хранятся. i-я строка матрицы представлена вектором TVector с StartIndex = i и Size = размер матрицы.

Методы класса

1. Конструктор TMatrix(int s = 10)

Параметры:

s – размер матрицы (по умолчанию 10)

Функционал: Создает верхнетреугольную матрицу размера s×s. Инициализирует базовый класс TVector размером s и StartIndex = 0. Для каждой строки i создает вектор TVector(s, i), где i - номер строки (StartIndex iй строки). Сложность: O(s)

2. Конструктор копирования TMatrix(const TMatrix& mt)

Параметры:

mt - матрица-источник

Функционал: Создает глубокую копию матрицы, вызывая конструктор копирования базового класса TVector. Сложность: $O(s^2)$, где s - размер матрицы

3. **Конструктор преобразования типа TMatrix(TVector<TVector<ValType>>& mt_vec)**

Параметры:

mt_vec - вектор векторов для преобразования в матрицу

Функционал: Создает матрицу из вектора векторов.

Сложность: O(s), где s - размер матрицы

4. Оператор сравнения operator==(const TMatrix& mt) const

Параметры:

mt - матрица для сравнения

Возвращаемое значение: bool - результат сравнения

Функционал: Проверяет равенство двух матриц, сравнивая векторы.

Сложность: $O(s^2)$, где s - размер матрицы

5. Оператор неравенства operator!=(const TMatrix& mt) const

Параметры:

mt - матрица для сравнения

Возвращаемое значение: bool - результат сравнения

Функционал: Возвращает отрицание результата operator==

Сложность: $O(s^2)$ - вызывает operator==

6. Оператор присваивания operator=(const TMatrix& mt)

Параметры:

mt - матрица-источник

Возвращаемое значение: TMatrix& - ссылка на текущий объект

Функционал: Выполняет копирование матрицы. Проверяет самоприсваивание. Делегирует работу оператору присваивания базового класса.

Сложность: $O(s^2)$, где s - размер матрицы

7. Оператор сложения operator+(const TMatrix& mt)

Параметры:

mt - матрица для сложения

Возвращаемое значение: TMatrix - новая матрица-результат

Функционал: Складывает две матрицы поэлементно. Матрицы должны иметь одинаковый размер. Использует операторы сложения векторов для каждой строки.

Сложность: $O(s^2)$, где s - размер матрицы

8. Оператор вычитания operator-(const TMatrix& mt)

Параметры:

mt - матрица для вычитания

Возвращаемое значение: TMatrix - новая матрица-результат

Функционал: Вычитает матрицу mt из текущей матрицы поэлементно.

Матрицы должны иметь одинаковый размер. Сложность: $O(s^2)$, где s - размер матрицы

9. Оператор ввода operator>>

Параметры:

in - входной поток

mt - матрица для заполнения

Возвращаемое значение: std::istream& - ссылка на поток

Функционал: Читает матрицу из потока построчно, используя оператор ввода для векторов.

Сложность: $O(s^2)$, где s - размер матрицы

10. Оператор вывода operator<<

Параметры:

out - выходной поток mt

- матрица для вывода

Возвращаемое значение: std::ostream& - ссылка на поток

Функционал: Выводит матрицу в поток построчно, используя оператор вывода для векторов. Каждая строка выводится с новой строки.

Сложность: $O(s^2)$, где s - размер матрицы

Краткие комментарии к тестам

Тест TestNameVector_Initialisation , TestNameVectorCheckConstructors

Проверяет корректность конструкторов TVector. Осуществляется проверка броска исключения при недопустимых параметрах (отрицательный размер, StartIndex превышающий размер, размер, превышающий MAX_VECTOR_SIZE). Также проверяется успешное конструирование при валидных значениях.

ТестTestNameVector_Operator_GetElement, TestNameSquareBrackets

Проверяет оператор []. Осуществляется доступ к элементам вектора, ожидаемый выброс при выходе за границы и при обращении к

неинициализированным позициям (pos < StartIndex). Сравнение с ожидаемыми значениями подтверждает корректность доступа. **Тест**

Тест TestNameVector_Operator_Equation, TestNameEquations

Проверяет операторы равенства, копирование, присваивание и само-присваивание. Сравниваются два одинаковых вектора, проверяется копирование, присваивание и корректность операторов == и !=.

Тест TestNameVector_Operator_Addition_Subtraction, TestNameVectorAddSub

Проверяет векторные операции: сложение, вычитание и умножение на скаляр. Тестируются корректность операторов +, -, * для векторов с одинаковым размером и ожидаемые результаты.

Тест TestNameVector_Operator_Multiplication, TestNameVectorMult

Проверяет скалярное произведение. Проверяется точность расчёта скалярного произведения двух векторов.

Тест TestNameVector_Operator_Add_Sub_Mul_Different_DImensions, TestNameVectorDifferentDImensions

Проверяет операции с разными длинами. Удостоверяется, что при попытке выполнить операции над векторами разных размеров генерируется исключение.

Тест TestNameVector_Operator_Add_Sub_Mul_Different_StartIndices, TestNameVectorDifferentStart

Проверяет операции с разным StartIndex. Проверяется, что операции над векторами с разным стартовым индексом работают корректно и дают ожидаемые значения.

Тест TestNameVector_Operators_Scalar_Operations, TestNameScalarOperators

Проверяет операции со скалярами. Тестируются корректность +=, -=, *= с скалярами и проверяется, что элементы до StartIndex (нулевые) корректно обновляются.

Тест TestNameMatrices_Initialisation, TestNameMatrixConstructors

Проверяет конструкторы TMatrix. Осуществляется проверка броска исключения при недопустимых размерах матрицы, успешное создание, копирование матрицы и конверсию из TVector<TVector<ValType>>.

Отклоняется, если строки не удовлетворяют условиям верхнетреугольной матрицы.

Тест TestNameMatrices_Operator_Equation, TestNameEquation

Проверяет операторы равенства/неравенства. Сравниваются матрицы, проверяется ==, !=, а также корректность присваивания матрицы самой себе.

Тест TestNameMatrices_Operator_Addition_Subtraction, TestNameAddSub

Проверяет сложение и вычитание матриц. Тестируются корректность операторов + и - для матриц одинакового размера, а также проверяется выброс при попытке сложить/вычесть матрицы разного размера.