Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Нижегородский Государственный Университет им.

Н.И.Лобачевского» (ННГУ)

Институт информационных технологий математики и механики

Отчёт по лабораторной работе

Вектора и матрицы

Выполнил:

студент группы 3821Б1ФИ3

Исаев Д.А.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г.

Нижний Новгород

2022 г.

Содержание

[1. Введение 3](#_Toc104143053)

[2. Постановка задачи 4](#_Toc104143054)

[3. Руководство пользователя 5](#_Toc104143055)

[4. Руководство программиста 7](#_Toc104143056)

[a. Описание структуры программы 7](#_Toc104143057)

[b. Описание структур данных 7](#_Toc104143058)

[c. Описание алгоритмов 7](#_Toc104143059)

[5. Эксперименты 9](#_Toc104143060)

[6. Заключение 10](#_Toc104143061)

[7. Литература 11](#_Toc104143062)

# Введение

**C++** (читается *си-плюс-плюс*) — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

**C++** широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Изначально одним из принципов разработки было сохранение совместимости с C. Тем не менее C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

# Постановка задачи

Написать классы для работы с векторами и матрицами, использовать шаблоны.

Вектора в математическом понимании: имеется набор значений из N мерного пространства, размерность задается как параметр.

Матрица должна быть наследником вектора.

Классы вектора и матрицы должны быть вынесены в статическую библиотеку.

Продемонстрировать их работу на примере (написать в main пример).

Должны быть:

конструкторы (по умолчанию, инициализатор, копирования), деструктор, доступ к защищенным полям;

перегруженные операции: +, -, \*, /, =, ==, [], потоковый ввод и вывод;

перегруженные операции +, -, \*, / должны быть реализованы для векторов (вектор +-\*/ вектор), матриц (матрица +-\* матрица), матрично-векторные (матрица \* вектор и наоборот);

Дополнительные задания:

* Написать метод, ищущий количество вхождений указанного значения;
* Написать метод, ищущий все вхождения заданного значения;
* В классе вектор должна быть возможность отсортировать его тремя способами (пузырек, вставка, быстрая сортировки).

# Руководство пользователя

Для созданий вектора есть три конструктора: по умолчанию, инициализатор, копирования (рис. 1).

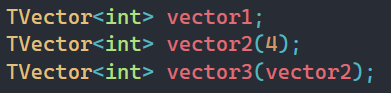


Рисунок 1. Конструкторы векторов

Можно складывать, вычитать, умножать, делить, присваивать, сравнивать вектора и обращаться отдельно к элементам вектора (рис. 2). Есть доступ к полю len - количество элементов в векторе.

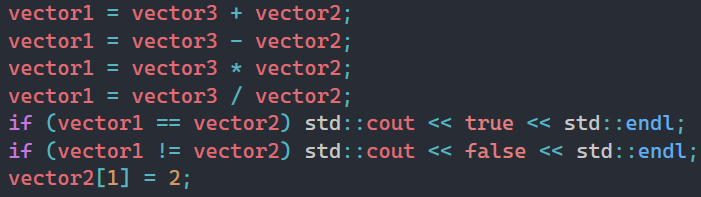


Рисунок 2. Операции над векторами

Векторы можно выводить и задавать с клавиатуры. Если вы хотите ввести вектор с клавиатуры, то сначала надо указать его длину (рис. 3).

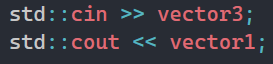


Рисунок 3. Ввод и вывод векторов

Есть дополнительные методы для работы с векторами: подсчет количества элементов определенного значения в векторе (или в матрице), нахождение всех индексов элементов заданного значения (также метод доступен для матриц), 3 вида сортировки (пузырьком, вставками, быстрая) (рис. 4).

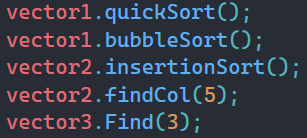


Рисунок 4. Дополнительные методы

Для созданий матрицы есть четыре конструктора: по умолчанию, инициализатор (создает квадратную / прямоугольную матрицу), копирования (копирует матрицу) (рис. 5).

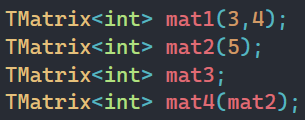


Рисунок 5. Конструкторы матриц

Можно складывать, вычитать, умножать, присваивать, сравнивать матрицы и обращаться отдельно к элементам матрицы (рис. 6). Есть доступ к полю width - количество векторов в матрице. Также можно умножить матрицу на вектор и наоборот (рис. 7).

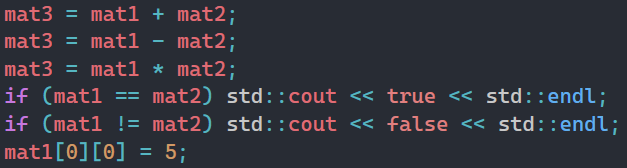


Рисунок 6. Операции с матрицами



Рисунок 7. Умножение вектора на матрицу и наоборот

Матрицы можно выводить и задавать с клавиатуры. Если вы хотите ввести матрицу с клавиатуры, то сначала надо указать её ширину, а затем длину (рис. 8).



Рисунок 8. Ввод и вывод матриц

# Руководство программиста

* 1. Описание структуры программы

Заголовочный файл Vector.h состоит из шаблонного класса TVector и реализации методов класса.

Заголовочный файл Matrix.h является наследником класса TVector содержит методы для работы с матрицами.

main - программа запрашивает выбор типа чисел: целые или дробные. При выборе соответствующего типа демонстрируются примеры методов.

* 1. Описание структур данных

Класса TVector имеет 2 поля len и data, где находятся длинна массива и массив шаблонных элементов соответственно.

Класса TMatrix имеет 3 поля width, len, data. data содержит вектор векторов, width количество векторов в основном векторе, len длина внутреннего вектора.

* 1. Описание алгоритмов

ВЕКТОР. Конструктор по умолчанию зануляет 2 поля вектора. Конструктор инициализатор принимает длину вектора, затем создает неинициализированный массив полученной длинны. Конструктор копирования копирует поля принимаемого вектора. Деструктор вызывается автоматически при выходе объекта из области действия или явно уничтожена вызовом delete. Метод GetLen возвращает значение поля len. При выполнении операторов +, -, \*, / у 2 векторов сравнивается длина, если их длины равны, то результатом будет вектор той же длины, полученный применением соответствующих операций поэлементно. Оператор присваивания вектора - копирует поля вектора. Сравнение векторов происходит поэлементно. Оператор [] принимает целочисленный индекс, если он не выходит за границы массива, то возвращает элемент, находящийся на указанной позиции. Метод Resize изменяет размер вектора на принимаемый целочисленный параметр. Метод findCol принимает параметр, который является элементом подсчета в векторе и возвращает количество вхождений этого элемента в векторе. Метод Find принимает параметр, который является элементом поиска в векторе. Данный метод возвращает массив индексов, на местах которых находится элемент поиска. Оставшиеся 3 метода сортируют массив в векторе (bubbleSort, quickSort, insertionSort).

МАТРИЦА. Конструктор по умолчанию зануляет 3 поля матрицы. Конструктор инициализатор создает квадратную матрицу (массив массивов), если принимаемый целочисленный параметр единственный или прямоугольную матрицу, если принимаемых параметров 2, тогда прямоугольная матрица будет длины первого параметра и ширины второго параметра. Конструктор копирования копирует поля полученной матрицы или вектора векторов. Деструктор вызывается автоматически при выходе объекта из области действия или явно уничтожена вызовом delete. Метод GetWidth возвращает значение поля width. При выполнении операторов +, - у 2 матриц сравниваются длины и ширины, если их длины и ширины равны, то результатом будет матрица той же длины и ширины, полученная применением соответствующих операций поэлементно. При выполнении оператора \* 2 матриц сравниваются длина первой и ширина второй, если их длина и ширина равны, то результатом будет матрица длины второй и ширины первой, полученная применением соответствующих операций поэлементно. Оператор \* дополнительно перегружен для умножения матрицы на вектор (и наоборот). Если длина матрицы равна длине вектора, то результатом является вектор, длина которого равна ширине матрицы, полученный как поэлементная сумма строк матрицы на вектор (и наоборот для вектора на матрицу). Оператор присваивания матрицы - копирует поля матрицы или вектора векторов. Сравнение матриц происходит поэлементно. Оператор [] принимает целочисленный индекс, если он не выходит за границы, то возвращает вектор, находящийся на указанной позиции. Метод Resize изменяет размер матрицы на 2 принимаемых целочисленных параметра (длина и ширина). Метод findCol принимает параметр, который является элементом подсчета в матрице и возвращает количество вхождений этого элемента в матрице. Метод Find принимает параметра, который является элементом поиска в векторе. Данный метод возвращает двумерный массив индексов, на местах которых находится элемент подсчета.

# Эксперименты

Проведем несколько тестов с различным количеством элементов для операций. Запишем время работы.

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | 10 000 | 1 000 000 | 100 000 000 |
| Вектор + вектор, сек | 0.001 | 0.014 | 1.587 |
| Матрица + матрица, сек | 0.001 | 0.025 | 3.36 |
| Матрица \* матрица, сек | 0.01 | 16.399 | ~ |

Таблица 1. Сравнение времени работы сортировок

# Заключение

В ходе лабораторной работы была написана статическая библиотека для работы с векторами и матрицами на языке С++. Реализованы все необходимые методы. Я убедился, что время выполнения операции + для 2 векторов O(n), у операции + для 2 матриц имеет сложность O(), а \* для 2 матриц O() (если матрицы квадратичные).

Также были выполнены дополнительные задания:

* Метод, ищущий количество вхождений указанного значения;
* Метод, ищущий все вхождения заданного значения;
* В классе вектор должна быть возможность отсортировать его тремя способами (пузырек, вставка, быстрая сортировки).

# Литература

1. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003. —461 с: ил.
2. C++. – Режим доступа: <https://ru.wikipedia.org/wiki/C%2B%2B>. – Загл. с экрана