Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Факультет вычислительной математики и кибернетики

Отчёт по лабораторной работе

Матрицы и векторы

Выполнил:

Назаров Н.С.

Проверил:

Нижний Новгород

2020 г.

Содержание

[Введение 3](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962758)

[Постановка задачи 4](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962759)

[Руководство пользователя 5](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962760)

[Руководство программиста 6](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962761)

[Описание структур данных 6](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962762)

[Описание алгоритмов 6](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962763)

[Описание структуры программы 6](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962764)

[Заключение 7](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962765)

[Литература 8](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962766)

[Приложения 9](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962767)

[Приложение 1 9](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962768)

[Приложение 2 9](file:///D:\загрузки\Шаблон%20отчёта%20по%20Лабораторной%20работе%20(2).doc#_Toc270962769)

# Введение

**Матрицы** широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений.

# 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. |
|  | 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); |
|  | в классе вектор должна быть возможность отсортировать его тремя способами (пузырек, вставка, быстрая сортировки, см. вторая лабораторная первого семестра). |

# Руководство пользователя

1. Создать переменную типа Matrix и Vector
2. Используя все доступные поля и методы, а также доступ к закрытым полям, написать программу для работы с классами
3. Получить результат

# Руководство программиста

## Описание структуры программы

Пять заголовочных файлов: Matrix.h, Matrix.cpp, Vector.h, Vector.cpp, main.cpp

Первый и третий для создания прототипов функций, второй и четвертый для описаний этих функций, пятый для работы со всеми данными нашей программы

## Описание структур данных

Закрытые поля, конструкторы по умолчанию, копирования, инициализации. Несколько методов для перегрузки операторов, выделения динамической памяти и создания матриц и векторов.

# Эксперименты

Сумма матриц : template <typename Type>

Matrix<Type> Matrix<Type>:: operator+ (const Matrix<Type>& a) {

if (this->rows = a.rows && this->cols = a.cols) {

Matrix<Type> NewMatrix(int rows, int cols);

for (int i = 0; i < this->rows; i++) {

for (int j = 0; j < this->cols; j++) {

NewMatrix[i][j] = this->m[i][j] + a.m[i][j];

}

}

return NewMatrix;

}

else cout << "Error operator +" << endl;

}

Сумма матриц на 100 элементов занимает 0.01 сек с погрешностями, а на 1000 уже 0.82 сек, разница примерно в 100 раз.

Умножение матриц: template <typename Type5>

Matrix<Type5> Matrix<Type5>:: operator\* (const Matrix<Type5>& a) {

if (this->rows = a.rows && this->cols = a.cols) {

Matrix<Type5> NewMatrix(rows, cols);

Type5 Sum;

for (int i = 0; i < this->rows; i++) {

for (int j = 0; j < this->cols; j++) {

Sum = 0;

for (int l = 0; l < this->rows; l++) {

Sum = Sum + (this->m[i][l]) \* (a.m[l][j]);

}

NewMatrix[i][j] = Sum;

}

}

return NewMatrix;

}

else cout << "Error operator \*" << endl;

}

Умножение матриц на 100 элементов занимает 0.01 сек с погрешностями, а на 1000 уже 8.2 сек, разница примерно в 1000 раз.

Умножение вектора и матрицы: template <typename Type36>

Type36 Vector<Type36> :: operator\* (const Vector<Type36>& b) {

if (this->GetCols2() == b.GetCols2()) {

Type36 Vec = 0;

for (int i = 0; i < b.GetCols2(); i++)

Vec = Vec + this->GetVector()[i] \* b.GetVector[i];

return Vec;

}

else cout << "Error operator \* vector" << endl;

}

Умножение матрицы на 100 элементов занимает 0.000006 сек с погрешностями, а на 1000 уже 0.005 сек, разница примерно в 1000 раз.

Для BubbleSort разница 1000, для InsertionSort 1000, для QuickSort 20

# Заключание

В результате моей работы, мне удалось самостоятельно разобраться в основах темы: «Матрицы и операции над ними», курса Высшей Математики. Прочитав множество источников, просмотрев множество сайтов, мне удалось систематизировать основные данные по этой теме. Очень надеюсь, что моя работа может помочь разобраться другим ученикам.