

# Библиотека классов для работы с разреженными матрицами с «длинной» арифметикой

Алексей Сальников

2016

## 1. Введение

Разреженные матрицы — матрицы, где большая часть элементов имеет нулевые значения. В связи с этим, несмотря на то, что матричные операции универсальны, для своей работы они требуют несколько других форматов хранения данных и других алгоритмов реализации матричных операций. В большей степени это скорее работа со списками, нежели с двумерным массивом.

Длинная арифметика — арифметика с числами не помещающимися в стандартные типы данных, например `long long int`. Поэтому работа с ними, это скорее операции со строками.

В рамках данного задания требуется написать 2 серии программ:

1. Генераторы матриц и векторов. Матрицы и векторы записываются в текстовые файлы в определённом формате. Генераторы должны уметь генерировать объекты удовлетворяющие нужным свойствам.
2. Решатели. Программы осуществляющие набор операций над матрицами. В основном предполагается, что это программы тестирующие библиотеку и иллюстрирующие её возможности.

## 2. Структура библиотеки

### 2.1. основные Классы

Библиотека должна включать в себя следующие классы:

- Класс **Rational\_number** — предназначен для хранения рациональных чисел и операций с ними. Напоминаю, что в рациональных числах числитель целый, а знаменатель натуральный. В случае необходимости хранения 0, считать, что и в числителе и в знаменателе написан 0, и это единственная ситуация, когда 0 написан в знаменателе. Все целые числа (кроме 0) должны в знаменателе хранить 1. Предусмотреть метод, позволяющий привести число к несократимой дроби. При этом сами значения чисел должны храниться как строки `char*`.
- Класс **Vector** — хранит разреженный вектор рациональных чисел. Рациональные числа собраны в абстрактную структуру данных словарь (dictionary), ключом в ней является координата в векторе.
- Класс **Matrix** — хранит разреженную матрицу. Данные хранятся аналогично вектору, за исключением того, что ключом является пара координат.
- Набор классов для обработки возникающих исключительных ситуаций, имеющих место при работе с векторами, матрицами и числами. Классы должны позволять печатать осмысленные сообщения об ошибках включая возможность напечатать объекты при работе с которыми возникли ошибки. например при операции деления напечатать оба операнда.

## 2.2. Рациональные числа

Для чисел должны быть переопределены корректным образом все стандартные операции: `+`, `-`, `*`, `/`, `<`, `<=`, `>`, `>=`, `==`, `!=`, `++`, `--`, `+=`, `-=`, `*=`, `/=`. Операции с рациональными числами так же должны работать, когда в качестве одного из операндов указан целочисленный базовый тип данных например `unsigned short`. Также должно быть определено преобразование в `int`, `long int`, `short` — при этом если преобразование таково, что значение выходит за диапазон или у числа есть дробная часть, то необходимо генерировать исключение и не производить операцию.

Также определить методы: `round`, `floor`. Должны присутствовать оператор присваивания, а также конструкторы: по умолчанию, из числа типа `long int`, из двух строковых констант (типа `char*`), из одной строковой константы. Для приведения числа к каноническому виду создать метод `make_canonical()`.

## 2.3. Вектора

Для векторов необходимо реализовать операции сложения/вычитания, умножения/деления на рациональные числа. А также умножения на матрицу. Необходимо реализовать операции сложения и вычитания двух векторов, а также унарный минус.

Необходимо переопределить `[]` для доступа к элементу вектора (в случае выхода за диапазон генерировать исключение).

Должен присутствовать конструктор, позволяющий создать вектор по имени файла. Должен присутствовать метод `write`, который записывает вектор в файл по имени файла. Необходим мы конструкторы которые заполняют вектор нулями, а так же единицами. Не забыть про конструктор копирования и оператор присваивания.

## 2.4. Матрицы

Для матриц определить операции `+`, `-`, `*` и унарный минус. Конструкция побитового отрицания должна генерировать транспонированную матрицу. Операция `[]` должна быть определена для нескольких типов (`Matrix_coords`, `Matrix_row_coord`, `Matrix_column_coord`), при этом для координат возвращается элемент, а для строки и столбца вектор.

Матрица должна уметь читаться и записываться в файл. Предусмотреть конструкторы которые будут создавать матрицу заданной размерности заполненную нулями, заполненную единицами. Конструктор который создаёт единичную матрицу. Не забыть про конструктор копирования и оператор присваивания.

## 2.5. Некоторые общие моменты

Для всех классов должны быть предусмотрены методы позволяющие явно задавать поля. Например в матрице должен быть метод позволяющий в позицию  $(i, j)$  положить значение.

Для рациональных чисел, с точки зрения экономии памяти рационально хранить счётчик ссылок на объект и в некоторых случаях вместо копирования строк, составляющих рациональное число изменить счётчик ссылок, и память не копировать.

Для каждого класса описанного выше должен быть определён метод `to_string()`, который превращает этот объект в текст (тип `char *` с переносами строк внутри неё.) выделяя при этом для данного текста память.

Для вектора координаты и значения удобно хранить в виде сбалансированного бинарного дерева поиска. Однако его реализация несколько трудоёмка и достаточно будет решения в виде динамического массива классов. В случае с матрицами необходимо использовать либо хеш таблицы, либо вводить отношение больше для сравнения пар координат матрицы.

При реализации данного задания **запрещается пользоваться STL.**

### 3. Форматы файлов

Файлы для хранения векторов и матриц – текстовые. В них могут встречаться комментарии. Комментарий начинается символом ‘#’.

Файл для хранения векторов начинается со слова *vector* далее пробел, далее размерность вектора. Затем следуют координаты и числа. каждое число с координатами на своей строке. сперва в строке идёт координата в векторе, а затем после, произвольного количества пробельных символов рациональное число. В файле могут быть пустые строки. Координаты нумеруются с единицы.

Рационально число записано так: сперва идёт числитель, вместе со знаком, затем после символа ‘/’ знаменатель. В случае если число целое, то символ ‘/’ и знаменатель могут отсутствовать.

В случае матриц вместо слова *vector* идёт слово *matrix*, далее число задающее число строк матрицы, затем число столбцов матрицы. Далее с новой новой строчки значения матрицы. Формат значений следующий: сперва номер строки в матрице, потом номер столбца, далее число (как было описано ранее).

Пример файла с вектором:

```
#
# This file describes
# sparse vector
#
vector 50000

1      100
6000   23 / 5
7      -5/3
22     44 /1
```

Пример файла с матрицей:

```
#
# This file describes
# sparse matrix
#
matrix 50000 5000

1      1      100
6000   2      23 / 5
7      1      -5/3
22     2      44 /1
```