

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Московский Авиационный Институт  
(Национальный Исследовательский Университет)

Институт №8 "Информационные технологий и прикладная математика"  
Кафедра 806 "Вычислительная математика и программирование"

Курсовая работа  
по курсу "Вычислительные системы"  
2 семестр

Задание 7  
Разреженные матрицы

Автор работы:  
студент 1 курса, группы М8О-106Б-20  
Леухин М. В.  
Руководитель проекта:  
Дубини А. В.

Москва, 2021

# Содержание

<b>1</b>	<b>Постановка задачи</b>	<b>3</b>
<b>2</b>	<b>Основная часть</b>	<b>4</b>
2.1	Разреженные матрицы . . . . .	4
2.2	Описание алгоритма . . . . .	4
<b>3</b>	<b>Вывод</b>	<b>5</b>
<b>4</b>	<b>Список источников</b>	<b>5</b>

# 1 Постановка задачи

**Задание:** Составить программу на языке Си с функциями для обработки прямоугольных разреженных матриц с элементами целого типа, которая:

1. Вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате (по строкам), с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой.
2. Печатает введенные матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном (естественном) виде.
3. Выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путём обращения к соответствующим функциям
4. Печатает результат преобразования (вычисления) согласно заданной схеме размещения и в обычном виде.

В процедурах и функциях предусмотреть проверки и печать сообщений в случаях ошибок в задании параметров. Для отладки использовать матрицы, содержащие 5–10% ненулевых элементов с максимальным числом элементов 100.

**Вариант схемы размещения матриц:** один вектор. Ненулевому элементу соответствуют две ячейки: первая содержит номер столбца, вторая содержит значение элемента. 1"в первой ячейки означает конец строки, а вторая ячейка содержит в этом случае номер следующей хранимой строки. 1"в обеих ячейках являются признаком конца перечня ненулевых элементов разреженной матрицы.

**Преобразование:** Найти элемент матрицы, ближайший к заданному значению. Разделить на него элементы строки и столбца, на пересечении которых он расположен. Если таких элементов несколько, обработать все.

## 2 Основная часть

### 2.1 Разреженные матрицы

**Разреженная матрица** — это матрица с преимущественно нулевыми элементами (если же большая часть элементов ненулевые, матрицу называют **плотной**). Разреженные матрицы больших размеров часто возникают при решении таких задач, как дифференциальные уравнения в частных производных. При хранении и преобразовании разреженных матриц в компьютере бывает полезно (а часто и необходимо) использовать специальные алгоритмы и структуры данных, которые учитывают разреженную структуру матрицы. Операции и алгоритмы, применяемые для работы с обычными, плотными матрицами, применительно к большим разреженным матрицам работают относительно медленно и требуют значительных объёмов памяти. Однако разреженные матрицы могут быть легко сжаты путём записи только своих ненулевых элементов, что снижает требования к компьютерной памяти.

### 2.2 Описание алгоритма

**Цель:** написать функцию, выполняющую поставленную в введении задачу. Удобно будет разделить задание на 2 части: 1) поиск элемента (элементов) с ближайшим к заданному значением и 2) преобразование всех элементов строки и столбца, на пересечении которых он расположен.

**1) Нахождение элемента.** Здесь всё довольно просто — проходим по нашей матрице и рассматриваем каждый элемент. Заведём переменную  $dx$ , в которой будет храниться модуль разницы между заданным значением и последним найденным ближайшим к нему элементом. В начальный момент времени в переменной  $dx$  будем хранить значение  $DBL\_MAX$  из библиотеки `<float.h>`. Стоит учитывать тот факт, что значений, подходящих под условие, может быть два. Например, пусть дано значение  $n$  — тогда нам подходят все элементы со значением  $n - dx$  и  $n + dx$  при минимальном  $dx$  для этой матрицы (очевидно, что если в матрице есть элемент со значением  $n$ , то два этих значения совпадают). Чтобы не проходить по всей матрице 2 раза, изначально заведём массивы индексов — в них будем хранить индексы всех элементов, подходящих под условие. Если находим новый элемент такой, что новое значение  $dx$  меньше предыдущего — очищаем массив индексов и добавляем в него индекс нового элемента. В моём примере я использую 4 вектора для индексов: 2 вектора под индексы  $i$  и  $j$  всех элементов со значением  $n + dx$  —  $iidf$  и  $jidf$ , и 2 вектора под индексы  $i$  и  $j$  всех элементов со значением  $n - dx$  —  $iids$  и  $jids$ . Во время написания этого отчёта до меня дошло, что проще было бы завести некую структуру *Point* и хранить в ней два индекса — в таком случае понадобилось бы лишь 2 вектора структур *Point* под элементы вида  $n + dx$  и вида  $n - dx$ , что сделало бы структуру программы значительно проще. Но не судьба.

**2) Преобразование матрицы.** Здесь главная сложность состоит в том, что мы не имеем константного доступа к элементам строки/столбца, как в случае с вариантом хранения матрицы в виде двумерного массива. Поэтому проходим по всей матрице ещё раз и рассматриваем каждый элемент. Если очередной элемент имеет хотя бы один индекс, который есть в одном из векторов  $iidf$  или  $jidf$  — делим его значение на  $n + dx$ . Если хотя бы один его индекс есть в ветке  $iids$  или  $jids$  — делим его значение на  $n - dx$ .

### 3 Вывод

Использованные приёмы позволяют использовать значительно меньше памяти для хранения огромных разреженных матриц, а также достаточно быстро получать доступ к их элементам. Естественно, эффективность использования таких матриц обратно пропорционально зависит от заполненности — то есть чем меньше ненулевых элементов, тем лучше.

### 4 Список источников

- Методические указания к выполнению курсовых работ. Зайцев В. Е.
- Записи с семинара, посвящённого КП7. Дубинин А. С.