**Отчет по лабораторной работе №3**

**Тема: «Обработка разреженных матриц»**

Выполнила:

студентка ИУ7-35Б

Лаврова Анастасия

Вариант №11

Задача №6

**Цель работы:**

Реализовать алгоритмы обработки разреженных матриц, сравнить эффективность использования этих алгоритмов (по времени выполнения и по требуемой памяти) со стандартными алгоритмами обработки матриц при различном процентном заполнении матриц ненулевыми значениями и при различных размерах матриц.

**Условие задание:**

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов: - вектор A содержит значения ненулевых элементов; - вектор IA содержит номера строк для элементов вектора A; - связный список JA, в элементе Nk которого находится номер компонент в A и IA, с которых начинается описание столбца Nk матрицы A.

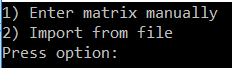
1.Смоделировать операцию умножения вектора-строки и матрицы, хранящихся в этой форме, с получением результата в той же форме. 2. Произвести операцию умножения, применяя стандартный алгоритм работы с матрицами. 3. Сравнить время выполнения операций и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.

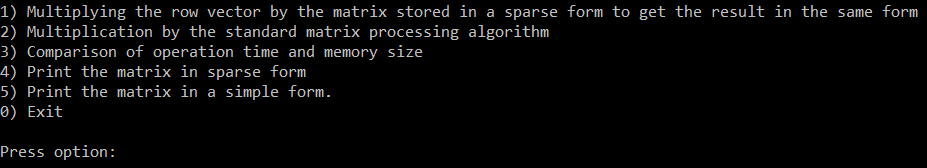
**Обращение к программе:**

Через консоль

**Техническое задание:**

Выбор способа ввода (вручную или считать из файла):

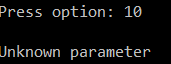




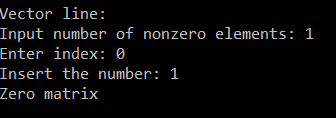
1. Умножение вектора строки на матрицу, хранящуюся в разреженной форме, чтобы получить результат в той же форме
2. Умножение по алгоритму стандартной матричной обработки
3. Сравнение времени работы и объема памяти
4. Печать матрицы в разреженной форме
5. Распечатайте матрицу в простой форме.

**Аварийные ситуации**:

Ввод неправильного параметра меню:



Получение нулевой матрицы:



Выход за границы допустимого диапазона индексов:  


Ввод несуществующего размера матрицы:  

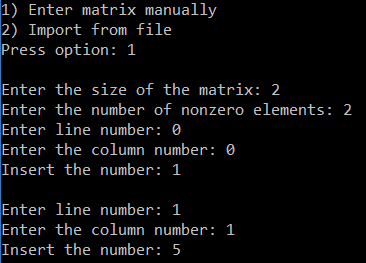

Некорректный ввод данных:

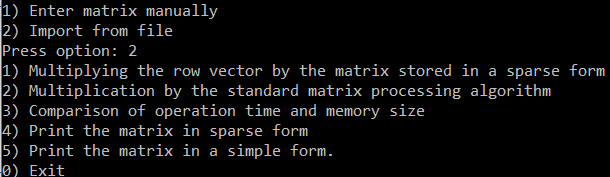


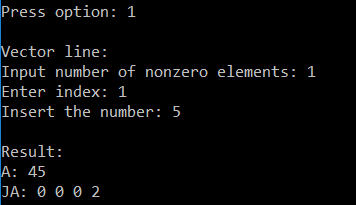
Печать в разреженной форме при отсутствии ненулевых элементов:

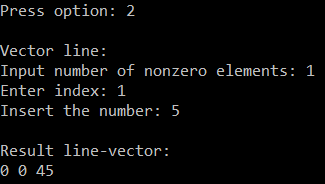


**Тесты:**  
Ввод значений (вручную и считать из файла)

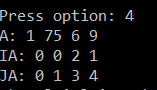


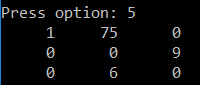


Умножение матрицы на строку-вектор в разреженной форме:  


Умножение в обычной форме:  


Напечатать матрицы в разреженном виде:



Напечатать матрицу в обычном виде:  


**Функции:**

int \*\* hand\_input(int \*size, int \*count)

Вход: размер массива и количество ненулевых элементов

Выход: сформированная матрица

int \*\* file\_input(int \*size, int \*count)

Вход: размер массива, количество ненулевых элементов, исходный файл

Выход: сформированная матрица

void inputing(int \*\*mtr, int size, int count, int \*A, int \*IA, int \*JA)

Вход: матрица, размер матрицы, количество ненулевых элементов, массивы A, JA, IA

Выход: заполненные массивы A, JA, IA

int menu\_1(void) (int menu\_2(void))

Вход: -

Выход: печать меню программы

int mult\_mtr(int \*\*matrix, int size)

Вход: матрица, размер матрицы

Выход: результат умножения в обычной форме матрицы и вектора-строки

int mult\_mtr\_raz(int \*A, int \*IA, int\*JA, int count, int size)

Вход: массивы A, IA, JA, количество ненулевых элементов, размер матрицы

Выход: результат умножения в разреженной форме матрицы и вектора-строки

void print\_mtr(int \*\*mtr, int size)

Вход: матрица, размер матрицы

Выход: печать матрицы на экран в обычной форме

void print\_mtr\_raz(int \*A, int \*IA, int\*JA, int count, int size)

Вход: массивы A, IA, JA, количество ненулевых элементов, размер матрицы

Выход: печать матрицы в разреженной форме

void efficient()

Вход: -

Выход: печать таблицы с данными об эффективности программы

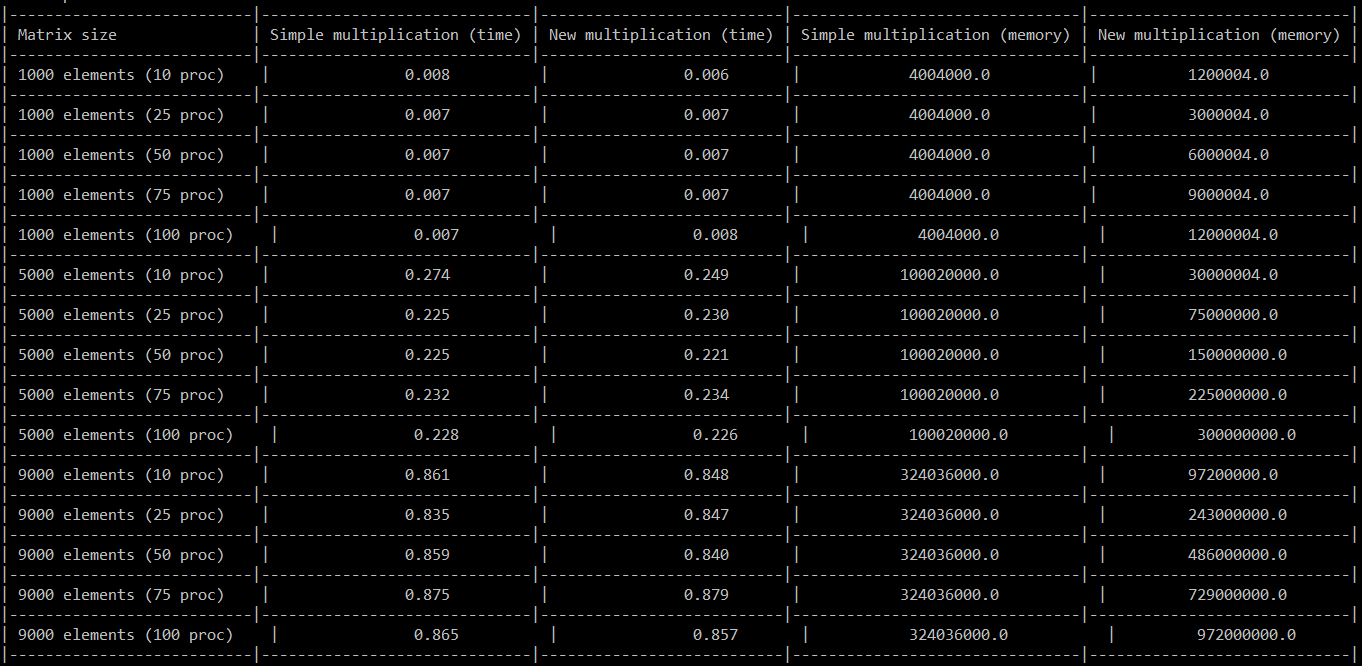
void calculation(int n, int p)

Вход: размер матрицы, процент наполненности матрицы

Выход: -

**Оценка эффективности:**

Были проведены тесты с матрицами, составленными из случайных чисел, размеры которых 1000, 5000, 9000 элементов. Под каждую матрицу память выделялась перед сортировкой и освобождалась после. Каждая матрица разной размерности заполнялась заново во избежание сортировки уже отсортированной матрицы.



Эффективность по времени:

Умножение матриц в виде 2 массивов и списка быстрее, чем умножение матриц обычным способом при наполненности матрицы менее 25% (а точнее при размерности матрицы 1000х1000 и наполненности менее 10%, при размерности матрицы 5000х5000 и наполненности менее 25%, при размерности матрицы 9000х9000 и наполненности менее 50%)

Эффективность по памяти:

Работа с двумя массивами и списком занимает меньше памяти, чем работа исключительно с матрицами если матрица заполнена не полностью.

**Ответы на вопросы:**

**1.** Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?

Разреженная – матрица, содержащая достаточно большое количество элементов, из которых лишь малая часть является ненулевыми (n^(1+g) для матрицы размерности n, g<1).

Простейшая схема хранения разреженной матрицы: хранить массив ненулевых элементов (AN), и два массива их «координат» (I, J) - номера столбцов и строк, в которых они расположены.

Кнут предложил дополнить эту схему также массивами NR (содержит номер из AN следующего ненулевого j элемента, расположенного в матрице по строке) и NC (номера –‘’- по столбцу), а также массивы JR и JC (указатели для входа в строку и столбец). Данная схема хранения избыточна, но позволяет легко осуществлять все матричные операции.

Чанг и Густавсон предложили схему разреженного строчного формата: хранятся массивы AN и J, а массив IA содержит номера (в AN) элементов, с которых начинается очередная строка матрицы.

**2.** Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?

Количество памяти, выделяемой под хранение обычной матрицы, определяется количеством её элементов (включая нулевые) и размером одного элемента, M \* N \* S. Память же под разреженную матрицу выдляется в зависимости от типа хранения. В то время как при формировании обычной матрицы выделяется один блок памяти на всю матрицу, при формировании разреженной памяти память выделяется по мере наполнения её ненулевыми элементами.

**3.** Каков принцип обработки разреженной матрицы?

Обработка разреженной матрицы предполагает работу только с ненулевыми элементами и зависит от схемы хранения матрицы, и типа операции.

**4.** В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?

Эффективность стандартных\разреженных алгоритмов обратно пропорциональна проценту «наполненности» матрицы. Чем больше в матрице ненулевых элементов, тем меньше выигрыш во времени и памяти; при превышении определённого уровня «наполненности» разреженные алгоритмы начинают давать даже более худшие результаты, нежели стандартные.