Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ЛАБОРАТОРНАЯ РАБОТА №3.

Студент Мицевич Максим Дмитриевич
фамилия, имя, отчество

Группа ИУ7-31Б

Студент _____ Мицевич М. Д.
подпись, дата *фамилия, и.о.*

Проверяющий _____ _____
подпись, дата *фамилия, и.о.*

Оценка _____

1. Описание условия задачи

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов: - вектор A содержит значения ненулевых элементов; - вектор JA содержит номера столбцов для элементов вектора A; - связный список IA, в элементе Nk которого находится номер компонент в A и JA, с которых начинается описание строки Nk матрицы A.
1. Смоделировать операцию умножения матрицы и вектора-столбца, 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.
4. Предоставить возможность ввода/вывода матриц в различных формах.

2. Техническое задание

1. Описание исходных данных

Все входные данные являются строками. В цикле вводятся с клавиатуру числа от 0 до 11, которые определяют дальнейшие поведение программы.

Формат ввода:

Возможные пункты меню, которые можно выбрать, введя число с клавиатуры:
0 — завершение программы
1 — генерирование файла с матрицей и чтение данных в матрицы из этого файла
2 — генерирование файла с столбцом и чтение данных в столбцы из этого файла
3 — печать матрицы в обычном виде
4 — печать столбца в обычном виде
5 — печать матрицы в виде 3 объектов
6 — печать столбца в виде 2 объектов
7 — чтение матрицы с консоли
8 — чтение вектора с консоли
9 — обычное умножение матрицы на вектор
10 — умножение разреженной матрицы на разреженный вектор
11 — получение результатов по времени
При чтение матрицы (вектора) из сгенерированного файла вводится размер матрицы (вектора), минимальное число в матрице (векторе), максимальное число в матрице (векторе), а также процент разреженности матрицы (вектора). При чтении матрицы (вектора) из стандартного потока ввода вводится размер матрицы (вектора) и ее (его) элементы. Также возможен координатный ввод матрицы и вектора, когда пользователь вводит координаты ненулевых элементов и их значения.

Ограничения:

- Минимальное число в генерируемой матрице не может быть равно 0
- Максимальное число в генерируемой матрице не может быть равно 0
- Количество столбцов в матрице должно совпадать с количеством строк в векторе-столбце

2. Описание результата программы

При корректных входных данных будет выполнен один из пунктов меню, иначе выведено сообщение об ошибке.
Формат вывода:
Обычная матрица выводится построчно. Матрица, хранящаяся в виде трех объектов выводится следующим образом: сначала печатается вектор с значениями ненулевых элементов матрицы, затем печатается вектор с номерами столбцов, в которых хранятся элементы из вектора значений, затем выводится связанный

список, в котором хранятся индексы элементов из первых двух векторов, с которых начинаются соответствующие строки.
Вектор-столбец выводится поэлементно на каждой строке. Вектор-столбец в виде двух объектов выводится следующим образом: сначала выводится на экран вектор с значениями ненулевых элементов в векторе-столбце, затем выводится вектор, в котором содержатся номера строк соответствующих элементов из вектора значений.
Таблица с временем работы содержит три столбца: процент нулей в матрице, время работы стандартного метода в секундах, время работы метода для разреженных матриц в секундах. Также перед таблицей печатается размер матрицы, для которой выполняется умножение.
Ограничения:

- Время работы сортировки содержит максимум 7 значащих цифр

3. Описание задачи, реализуемой программой
Программа выполняет умножение матрицы на вектор-столбец, хранящихся в разный форматах. Возможен ввод матрицы и вектора-столбца и их вывод в разных форматах.

4. Способ обращения к программе
Обращение к программе происходит с помощью стандартных потоков ввода и вывода.

5. Описание возможных аварийных ситуаций и ошибок пользователя
Ошибки операционной системы при работе с динамической памятью и файлами. Выход за ограничения минимального значения, максимального значения, размера матрицы (вектора) или процента нулей в матрице (векторе) при ее (его) генерации. Неправильный размер или значения матрицы (вектора) при ее (его) в вводе из стандартного потока ввода. Во всех указанных случаях программа завершится корректно и сообщит об ошибке.

3. Описание внутренних структур данных

Для хранения обычных матриц была выбрана данная структура:
typedef int elem_t;

```
typedef struct
{
   size_t rows;
   size_t columns;
   elem_t **matrix;
}matrix_t;
```

поле rows отвечает за количество строк в матрице, поле columns за количество столбцов в матрице, matrix – указатель на динамическую матрицу.
В динамической матрице отдельными блоками выделяются массив указателей на строки и каждая строка.

Для хранения матрицы в указанной форме используются данные структуры, они были выбраны в связи с тем, чтобы уменьшить количество занимаемой памяти и время обработки для разреженных матриц.

```
struct row_start
{
   size_t index;
   struct row_start *next;
};

typedef struct row_start row_start_t;

typedef int pelem_t;

typedef struct
{
   pelem_t *values;
   size_t *columns_numbers;
   size_t columns;
   size_t columns_allocated;
   row_start_t *head_row_starts;
}pmatrix_t;
```

values – вектор с значениями, column_numbers – номера столбцов элементов из первого вектора, columns – количество элементов в первых двух векторах, columns_allocated – количество элементов, под которые выделена память в первых двух векторах, head_row_starts – связанный список, в котором хранятся индексы

элементов из первых двух векторов, с которых начинаются соответствующие строки.

Вектор-столбец хранится аналогично матрице.
В нормальном виде:

```
typedef struct
{
   size_t rows;
   elem_t *column;
}column_t;
```

В виде двух объектов:

```
typedef struct
{
   pelem_t *values;
   size_t *columns_numbers;
   size_t columns;
   size_t columns_allocated;
}pvector_t;
```

Для матрицы 512*512 с 25%-ой заполненностью на 64-битной машине с int == 4 байта будет занята соответствующая память:
Матрица: 8*3 + 8*512 + 512*512*4 = 1029Кб
Указанный вид: 8*5 + 256*256*2*4 + 16*512 = 521Кб
Выигрыш в 2 раза.

4. Алгоритм

Алгоритм обычного умножения(матрица хранится по строкам):
1. Перемножаем попарно числа из строки матрицы и соответствующие им числа из столбца вектора и находим сумму этих перемножений.
2. Записываем получившееся число в результирующий вектор-столбец.
3. Переходим в следующую строку и повторяем пункты 1-2.

Алгоритм умножения:
1. Идем по каждому ненулевому элементу строки матрицы.
2. Проверяем есть ли в матрице столбце ненулевой элемент на нужной позиции, если такой есть, то умножаем его на элемент строки и находим сумму этих перемножений.
3. Записываем получившееся число(если оно не ноль) в результирующий вектор-столбец.
4. Повторяем 1- 3, пока не дойдем до конца связанного списка, в котором хранятся индексы начала строк.

5. Тесты

Позитивные тесты:

Входные данные	Что проверяем	Ожидаемый выходной результат
Выбор: обычное умножение 123 020 456 1 2 0	Обычный тест	5 4 14
Выбор: умножение методом для разреженной матрицы 123 020 456 1 2 0	Обычный тест	5 4 14 0 1 2
Выбор: генерация матрицы Размер 5 5 Минимальное число -10 Максимальное число 10 Процент нулей 50	Проверка на создание матрицы	Создан файл содержащий матрицу из случайных чисел от -10 до 10 и на 50% заполненный нулями

Негативные тесты:

Входные данные	Что проверяем	Ожидаемый выходной результат
Выбор: ввод матрицы Размер 1000000 1000000	Проверка на выделение памяти	Код возврата равен 10
Выбор: умножение матрицы на вектор Размер матрицы равен 10 5 Размер вектора равен 9	Проверка на возможность умножения	Wrong sizes
Выбор: ввод матрицы 2 2 f	Проверка на неправильность ввода	Wrong sizes

6. Функции

Для создания матрицы
int allocate_matrix(matrix_t *mtr, size_t n, size_t m)

Для освобождения матрицы
void free_matrix(matrix_t *mtr);

Аналогичные функции для матрицы в виде трех объектов
int allocate_rows_start(row_start_t **rs);
void free_pmatrix(pmatrix_t *pmtr);

Чтение и печать матрицы
int fread_matrix(matrix_t *mtr, FILE *src);
void fwrite_matrix(matrix_t mtr, FILE *dst);

Умножение матрицы на вектор обычным способом
int mlt_mtr_vec(matrix_t mtr, column_t column, column_t *res);

Чтение и печать матрицы в виде трех объектов
int read_pmatrix_from_matrix(pmatrix_t *pmtr, matrix_t mtr);
int fread_pmatrix(pmatrix_t *pmtr, FILE *src);
void fwrite_pmatrix(pmatrix_t pmtr, FILE *dst);

Умножение матрицы на вектор в виде трех объектов
int mlt_pmtr_pvec(pmatrix_t pmtr, pvector_t column, pvector_t *res);

7. Оценка эффективности программы.

Оценка сложностей:
1. Оценка обычного перемножения:
Два цикла с постоянными границами, следовательно O(n^2)

2. Умножение матрицы на вектор в виде трех объектов:
Два цикла: проход по строкам матрицы, проход по ненулевым элементам строки, т.е O(n^2).
3. Оценка времени и памяти:

```
matrix 200x200
 percent     normal    packed normal mem  packed mem
 10 0.000109s 0.000009s   161624b     35256b
 20 0.000105s 0.000025s   161624b     67256b
 30 0.000104s 0.000054s   161624b     99256b
 40 0.000105s 0.000093s   161624b    131256b
 50 0.000104s 0.000139s   161624b    163256b
 60 0.000107s 0.000157s  161624b    195256b
matrix 400x400
 percent    normal    packed normal mem  packed mem
         10 0.000416s 0.000045s   643224b    134456b
         20 0.000416s 0.000128s   643224b    262456b
         30 0.000417s 0.000254s   643224b    390456b
         40 0.000428s 0.000428s   643224b    518456b
         50 0.000413s 0.000595s   643224b    646456b
         60 0.000415s 0.000677s   643224b    774456b
matrix 800x800
    percent    normal    packed normal mem  packed mem
         10 0.001655s 0.000190s  2566424b    524856b
         20 0.001657s 0.000517s  2566424b   1036856b
         30 0.001651s 0.001017s  2566424b   1548856b
         40 0.001658s 0.001744s  2566424b   2060856b
         50 0.001661s 0.002398s  2566424b   2572856b
         60 0.001665s 0.002804s  2566424b   3084856b
```

8. Выводы

Данный вид хранения разреженных матриц дает выигрыш в памяти перед обычной матрицей, если количество нулей в матрице больше 50 процентов от числа элементов. Выигрыш в скорости наблюдается, когда количество нулей в матрице больше 60-ти процентов. Также стоит отметить, что выигрышь в скорости становится более значительным при увеличении размеров матрицы.

Ответы на контрольные вопросы:

1. Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?
Разреженная матрица – это матрица, содержащая большой процент нулей.
Матрицу можно хранить в виде массива массивов или с помощью различных представлений разреженной матрицы (например, в виде 3-ёх объектов).

2. Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?
На хранение разреженной матрицы память выделяется в соответствии с количеством ненулевых элементов. На хранение полной матрицы память выделяется в зависимости от количества строк и столбцов.

3. Каков принцип обработки разреженной матрицы?
Алгоритмы обработки разреженных матриц предусматривают действия только с ненулевыми элементами и, таким образом, количество операций будет пропорционально количеству ненулевых элементов.

4. В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?
При большом размере матрицы или при малом количестве нулевых элементов.