Министерство науки и высшего образования 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

Отчет по лабораторной работе №3 «ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ»

Студент Равашдех Фадей Хешамович

Группа ИУ7 – 35Б

Преподаватель Никульшина Т. А.

2023 г.

Содержание

Описание условия задачи

Разработать программу сложения разреженных матриц. Предусмотреть возможность ввода данных, как с клавиатуры, так и использования заранее подготовленных данных. Разреженные матрицы хранятся и выводятся в форме трех объектов. Для небольших матриц можно дополнительно вывести матрицу в виде матрицы. Величина матриц - любая. Сравнить эффективность (по памяти и по времени выполнения) стандартных алгоритмов обработки матриц с алгоритмами обработки разреженных матриц при различной степени разреженности матриц и различной размерности матриц.

Описание технического задания

Входные данные:

Файл с данными: в файле содержится матрица в координатном виде: в первой строке записан разме матрицы и количество ненулевых элементов. На последующих строках записаны индексы элемента и его значения.

Целое число: задаёт пункт меню.

Числовые данные: ввод матриц в координатном виде.

Выходные данные:

Матрица (в обычном и координатном формате), результат сложения матриц, сравнение эффективности алгоритмов.

Функции меню программы:

1. Информация о программе
2. Ввести первую матрицу
3. Ввести вторую матрицу

4. Вывести первую матрицу
5. Вывести вторую матрицу
6. Вывести результат сложения двух матриц стандартным алгоритмом
7. Вывести результат сложения двух матриц алгоритмом для разреженных матриц
8. Вывести результаты сравнения сложений матриц в стандартном виде и в виде разреженной матрицы
9. Выйти

Обращение к программе:

Запускается через терминал:

./app.exe [*имя_файла_с_матрицей] [имя_файла_с_матрицей]*

Аварийные ситуации:

Неверные аргументы командной строки.

Файл не найден или программа не имеет прав на чтение.

Некорректный ввод данных из файла.

Некорректное количество непустых элементов матрицы.

Некорректный размер матриц.

Ошибка выделения памяти.

Действие с введенным пунктом отсутствует.

Введен символ окончания ввода EOF. Завершение программы.

Некорректный ввод. Количество введенных символов равно нулю или превысило ожидаемое.

Матрица не прошла проверку на корректность.

Описание структур данных

```
// Структура стандартной матрицы
typedef struct
{
    int **matrix;   // Указатель на массив указателей на строки матрицы
    size_t n, m;    // Количество строк и столбцов соответственно
} matrix_t;

// Структура разреженной матрицы
typedef struct
{
    int *a;     // Вектор A ненулевых элементов
    size_t *ia; // Вектор IA номеров элементов с которых начинается
описание i-й строки
    size_t *ja; // Вектор JA номеров столбцов элементов вектора A
    size_t n, m;// Количество строк и столбцов соответственно
    size_t k;   // Количество ненулевых элементов
} sparse_matrix_t;
```

Описание функций

Функции главного файла программы:

```
/**
 * @brief Контроллер аргументов программы
 *
 * @param argc количество аргументов
 * @param argv аргументы
 * @param matrix1 первая матрица
 * @param matrix2 вторая матрица
 */
void arg_controller(int argc, char **argv, matrix_t *matrix1,
matrix_t *matrix2, sparse_matrix_t *sparse_matrix1,
sparse_matrix_t *sparse_matrix2);

/**
 * @brief Контроллер
 *
 * @param option выбранный пользователем пункт меню
 * @param filename имя файла
 * @param table таблица
 * @param key_table таблица ключей
 * @return int код возврата
 */
int controller(int option, matrix_t *matrix1, matrix_t *matrix2,
sparse_matrix_t *sparse_matrix1, sparse_matrix_t *sparse_matrix2);
```

Функции модуля ввода/вывода:

```
/**
 * @brief Чтение матрицы из файла
 *
 * @param filename имя файла
 * @param matrix матрица
 * @return int код возврата
 */
int read_matrix(char *filename, matrix_t *matrix);

/**
 * @brief Ввод матрицы
 *
 * @param matrix матрица
 * @return int код возврата
 */
int input_matrix(matrix_t *matrix);

/**
 * @brief Вывод матрицу в виде таблицы
```

```
 *
 * @param matrix матрица
 */
void print_matrix_as_table(matrix_t matrix);

/**
 * @brief Контроллер вывода матрицы
 *
 * @param matrix матрица
 */
void print_matrix(matrix_t matrix);

/**
 * @brief Вывод меню программы
 *
 */
void print_menu(void);

/**
 * @brief Вывод информации о программе
 *
 */
void print_info(void);

/**
 * @brief Чтение строки определенной длины из стандартного потока
ввода
 *
 * @param pch указатель на принимаемую строку
 * @param n длина строки
 * @return int код возврата
 */
int get_input(char pch[], size_t n);

/**
 * @brief Получение выбранного пользователем пункта
 *
 * @param option выбранный пункт
 * @return int код возврата
 */
int get_option(size_t *option);

/**
 * @brief Вывод сообщения, поясняющего ошибку
 *
 * @param rc код возврата
 */
void print_error_msg(int rc);

/**
```

```
 * @brief Вывод сообщения, пояснящего ошибку, и возврат того же
кода возврата
 *
 * @param rc код возврата
 * @return int код возврата
 */
int print_error(int rc);
```

Функции модуля для работы с матрицами:

```
/**
 * @brief Структура стандартной матрицы
 *
 */
typedef struct
{
    int **matrix;   // Указатель на массив указателей на строки
матрицы
    size_t n, m;    // Количество строк и столбцов соответственно
} matrix_t;

/**
 * @brief Структура разреженной матрицы
 *
 */
typedef struct
{
    int *a;     // Вектор A ненулевых элементов
    size_t *ia; // Вектор IA номеров элементов с которых
начинается описание i-й строки
    size_t *ja; // Вектор JA номеров столбцов элементов вектора A
    size_t n, m;// Количество строк и столбцов соответственно
    size_t k;   // Количество ненулевых элементов
} sparse_matrix_t;

// Объявления функций

/**
 * @brief Выделение памяти под матрицу
 *
 * @param n количество строк в матрице
 * @param m количество столбцов в матрице
 * @return int** указаетель на массив n указателей на начала строк
матрицы
 */
int **allocate_matrix(size_t n, size_t m);

/**
 * @brief Является ли разреженная матрица
 *
```

```
 * @param matrix
 * @return int
 */
int is_matrix_valid(matrix_t matrix);

/**
 * @brief Освобождение матрицы
 *
 * @param matrix матрица
 */
void free_matrix(matrix_t *matrix);

/**
 * @brief
 *
 * @param matrix1
 * @param matrix2
 * @param matrix_res
 * @return int
 */
int matrix_addition(matrix_t matrix1, matrix_t matrix2, matrix_t
*matrix_res);

/**
 * @brief Создание разреженной матрицы
 *
 * @param n количество строк
 * @param m количество стоблцов
 * @return sparse_matrix_t разреженная матрица
 */
sparse_matrix_t create_sparse_matrix(size_t n, size_t m);

/**
 * @brief Проверка разреженной матрицы на корректность
 *
 * @param matrix матрица
 * @return int код возврата
 */
int is_sparse_matrix_valid(sparse_matrix_t matrix);

/**
 * @brief Проверка матрицы на корректность
 *
 * @param matrix матрица
 * @return int код возврата
 */
void free_sparse_matrix(sparse_matrix_t *matrix);

/**
 * @brief Сложение разреженных матриц
 *
```

```
 * @param matrix1 первая матрица
 * @param matrix2 вторая матрица
 * @param matrix_res получившаяся матрица
 * @return int код возврата
 */
int sparse_matrix_addition(sparse_matrix_t matrix1,
sparse_matrix_t matrix2, sparse_matrix_t *matrix_res);

/**
 * @brief Создание разреженной матрицы на основе обычной
 *
 * @param matrix матрица
 * @return sparse_matrix_t разреженная матрица
 */
sparse_matrix_t matrix_std_to_sparse(matrix_t matrix);

/**
 * @brief Создание матрицы на основе разреженной матрице
 *
 * @param matrix разреженная матрица
 * @return matrix_t матрица
 */
matrix_t matrix_sparse_to_std(sparse_matrix_t matrix);

/**
 * @brief Сравнение алгоритмов сложения
 *
 * @param matrix1 первая матрица
 * @param matrix2 вторая матрица
 * @param sparse_matrix1 первая разреженная матрица
 * @param sparse_matrix2 вторая разреженная матрица
 * @return int код возврата
 */
int compare(matrix_t matrix1, matrix_t matrix2, sparse_matrix_t
sparse_matrix1, sparse_matrix_t sparse_matrix2);
```

Описанние алгоритма

1. Программа запускается либо без ключей, либо с одним – именем файла-матриц, либо с двумя – именемами файлов-матриц.

2. Происходит считывание данных из файлов в матрицу. Если считывание успешно, то выводится сообщение об успешности считывания. Если нет, то выводится сообщение об ошибке и сообщается о возможности ввести матрицы позже с клавиатуры.

3. Пользователю выводится меню программы и предлагается выбрать и ввести пункт меню с необходимым действием.

4. Если действие предполагает ввод дополнительных данных (например, ввод таблиц), то пользователю предлагается ввести эти данные.

5. Если действие предполагает вывод, то выводятся данные (матрица, таблица сравнения алгоритмов).

6. В случае возникновния аварийной ситуации, пользователь получит сообщение, раскрывающее суть ошибки.

7. Алгоритм сложения разреженных матриц: происходит итерация по строкам разреженной матрицы (по вектору IA), для каждой строки известны индексы столбцов элементов двух матриц, находящихся на данной строке. В каждой строке происходит итерация по двум векторам индексов столбцов: если индексы столбцов равны, то значения скадываются и, если сумма не равна нулю, записываются в матрицу и берутся следующие индексы столбцов, если индексы столбцов не равны, то записывается значение наименьшего из столбцов и берется следующий индекс столбца этой матрицы. После этого, если на какой-то строке не остается индексов столбцов в векторе одной из матриц, то тогда итерация продолжается по индексам столбцов вектора другой матрицы.

НАБОР ТЕСТОВ

№	Название теста	Ввод	Вывод
1	Некорректный запуск программы	./app.exe	«Ошибка: Неверные аргументы командной строки.» «Команда вызывается из командной строки так:\n./app.exe [matrix1.txt] [matrix2.txt]»
2	Некорректный ввод пункта меню	112	«Ошибка: Действие с введенным пунктом отсутствует.»
3	Некорректный ввод пункта меню	afaeFefWfcWV	«Ошибка: Действие с введенным пунктом отсутствует.»
4	Просмотр матрицы	3	*Матрица*
5	Просмотр матрицы, которой не существует	3	«Ошибка: Матрица не прошла проверку на корректность.»
6	Ввод матрицы	1 *данные матрицы*	«Матрица успешно введена.»
7	Некорректный ввод вод матрицы	1 -1	«Ошибка: Некорректная дата.»Введенное поле количество строк '-1' не прошло проверку. Поле должно иметь вид '^[1-9][0-9]{0,3}$'. Ошибка: Некорректный ввод. Количество введенных символов равно нулю или превысило ожидаемое.»
8	Сложение матриц, одна из которых некорректна	5	«Ошибка: Матрица не прошла проверку на корректность.»
9	Сложение матриц	5	*Матрица*

Оценка эффективности

Замеры времени (в секундах) выполнения сложения 1000 раз:

Процент заполнения	Размерность матрицы NxN	Время сложения обычных матриц	Время сложения разреженных матриц	На сколько % алгоритм разреженных матриц быстрее
100	1000	1.668964	16.696219	-90.004%
	100	0.021836	0.172553	-87.345%
	10	0.001195	0.006188	-80.688%
10	1000	1.658243	4.228947	-60.788%
	100	0.021870	0.040300	-45.732%
	10	0.001459	0.002440	-40.205%
5	1000	1.653384	2.068308	-20.061%
	100	0.022530	0.022951	-1.834%
	10	0.001307	0.001528	-14.463%
3	1000	1.651118	1.230433	34.190%
	100	0.023124	0.015178	44.406%
	10	0.001195	0.000876	36.416%
1	1000	1.643965	0.460828	256.742%
	100	0.022327	0.006006	271.745%
	10	0.001041	0.000536	94.216%

Замеры памяти в байтах двух матриц

Процент заполнения	Размерность матрицы NxN	Память матриц	Память разреженных матриц	На сколько % разреженные матрицы экономнее
100	10	848	2656	-68.072%
	100	80048	241696	-66.881%
	1000	8000048	24015784	-66.688%
50	10	848	1456	-41.758%
	100	80048	121696	-34.223%
	1000	8000048	12015952	-33.421%
30	10	848	976	-13.115%
	100	80048	73696	8.619%
	1000	8000048	7216024	10.865%
1	10	848	280	202.857%
	100	80048	4096	1854.297%
	1000	8000048	256096	3023.847%

Ответы на контрольные вопросы

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

Разреженная матрица – это матрица, большую часть которой занимают нули. Схемы хранения: обычная матрица; координатная запись; связная схема; “разреженный строчный формат”.

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

Память под обычную матрицу выделяется один раз размером N*M*sizeof(val_type), память под разреженную матрицу представляется динамической памятью и довыделяется при необходимости добавления элементов, в итоге общая память составляет K* (sizeof(val_type) + sizeof(size_t)) + N*sizeof(size_t), где K – количество ненулевых элементов, т.е. зависит от количества ненулевых элементов.

3. Каков принцип обработки разреженной матрицы?

Принцип заключается в итерации по строкам, где в каждой строке итерациция происходит по элементам строки от номера первого элемента строки до номера первого элемента следующей строки.

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

Применять стандартные алгоритмы обработки матриц эффективнее будет в случае, когда количество ненулевых элементов мало от общего числа элементов (для моего алгоритма <5%), т.е. зависит от количества ненулевых элементов матрицы.

Вывод

В результате выполнения этой лабораторной работы, я узнал о способах хранения разреженных матриц, научился использовать и складывать разреженные матрицы в сжатом виде, улучшил свои навыки работы с динамическими массивами и контролем динамически выделенной памяти.

В результате оценки эффективности, я пришёл к выводу, что представление матрицы в разреженном виде имеет смысл только при заполнении менее 5%. В ином случае, быстрее оказывается сложение обычных матриц. По памяти имеет смысл использовать разреженные матрицы при заполнении менее 30%.