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 Кладницкий А. Б.

Преподаватель

Группа ИУ7 – 32Б

2022 г.

1. **Описание условия задачи**

Разработать программу сложения разреженных матриц. Предусмотреть возможность ввода данных, как с клавиатуры, так и использования заранее подготовленных данных. Матрицы хранятся и выводятся в форме трех объектов. Для небольших матриц можно дополнительно вывести матрицу в виде матрицы. Величина матриц - любая (допустим, 1000*1000). 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.

2. **ТЗ**

   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.

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

   1. Исходные данные:
      - Файл, содержащий в себе матрицу, или введенная с клавиатуры матрица
      - Коды для управления меню
      - Другие данные, для работы с меню
      - Меню:
        0. Open file (if needed)
        1. Input matrix A (indexes)
        2. Input matrix A (normal form)
        3. Input matrix A (fill randomes)
        4. Read matrix A (indexes) from file
        5. Read matrix A (normal form) from file
        6. Read matrix A (fill randomes) from file
        7. Input matrix B (indexes)
        8. Input matrix B (normal form)
        9. Input matrix B (fill randomes)
        10. Read matrix B (indexes) from file
        11. Read matrix B (normal form) from file
        12. Read matrix B (fill randomes) from file
        13. Print matrix A

14. Print matrix A in raw form (3 arrays)
15. Print matrix B
16. Print matrix B in raw form (3 arrays)
17. Calculate summ of matrix A and B
18. Print result of summarizing of matrix A and B
19. Print result of summarizing of matrix A and B in raw form
20. Measuring
21. Print menu again

-1. Exit

2. Результирующие данные:
    - В зависимости от выполняемой задачи, вывод матрицы в заданном формате, вывод суммы матриц в заданном формате, результаты сравнительных тестов.
3. Задача программы:
    - Чтение матрицы из файла или из потока ввода в нужном формате
    - Вывод матрицы в нужном формате
    - Сложение матриц
    - Вывод результатов сравнительных тестов алгоритмов
4. Способ обращения к программе:
    - Запуск через терминал (./app.exe), работа через меню.
5. Возможные ошибки:
    - Ошибка работы с меню: неверный код
    - Ошибка работы с файлом: не существует или другая проблема
    - Ошибка выделения динамической памяти
    - Ошибка: нулевой указатель там, где он не ожидался
    - Неверный индекс
    - Несовпадение размеров матриц при сложении
    - Неверный ввод целого числа
    - Неверный ввод без знакового числа
    - Неверный ввод вещественного числа
    - Неподготовленные данные (сложение матриц при их отсутствии)

3. Описание внутренних структур данных

Тип данных matrix_t (разреженная матрица):

```
typedef struct matrix
{
  vect_t *data;
  vect_t *rows;
  list_t *lines;
  size_t m;
  size_t n;
} matrix_t;
```

Тип содержит в себе векторы data и rows, список lines и размеры матрицы.

```
typedef struct vect
{
  int *data;
  size_t len;
  size_t len_max;
} vect_t;
```

Тип vect_t содержит в себе указатель на динамически выделенную область для хранения значений data, размер текущей используемой области len и общий выделенный под нее размер len_max. Такая организация структуры помогает избежать постоянного использование функции realloc при добавлении нового элемента в вектор.

```
typedef struct list
{
  int value;
  struct list *next;
} list_t;
```

Тип list_t хранит в себе значения value узлов односвязного списка и ссылку на следующий узел (NULL, если конец).

Обычная матрица:

```
typedef struct mtr
{
  int *data;
  size_t m;
  size_t n;
} mtr_t;
```

Тип `mtr_t` содержит в себе ссылку на область памяти, в которой хранится матрица, и ее размерности.

4. **Алгоритм**
    1. Вывести меню
    2. Получить ответ от пользователя: какой пункт меню выполнять?
    3. Выполнять команды пользователя, пока не был введен -1 или достигнуто другое условие выхода из главного цикла.
    4. Сложение: добавление в результирующую матрицу элемента, если его не было в этой позиции, и увеличение элемента в данной позиции, если он был.
5. **Набор тестов**

Негативные тесты:

	Описание	Входные данные	Результат
1.	Некорректный пункт меню	-1, uhyv и т.д.	ERROR_BAD_KEY
2.	Несуществующий файл	non.txt	ERROR_BAD_FILE
3.	Файл без доступа	не хватает прав на взаимодействие	ERROR_BAD_FILE
4.	Нулевой размер матрицы	нулевой размер матрицы	ERROR_EMPTY_MTR
5.	Неверный индекс	i = hjwer4	ERR_BAD_INT
6.	Отрицательный индекс	i = -5	ERR_BAD_SIGN
7.	Индекс слишком большой	100 (матрица 10 на 10)	ERR_INDEX_OUT
8.	Неверное значение	qwik (не число)	ERR_BAD_INT
9.	Неверный «вес» для заполнения	testing	ERR_BAD_DOUBLE
10.	Ошибка выделения динамической памяти	???	ERR_ALLOC
11.	Неинициализированные матрицы	вызов функций вывода/сложения до ввода матриц	ERR_NEED_PREP

Позитивные тесты:

	Описание	Входные данные	Результат
1.	Открытие файла	название файла	Файл открыт и готов к использованию
2.	Чтение матрицы с клавиатуры	матрица по индексам	Матрица считана
3.	Чтение матрицы с клавиатуры	матрица в обычном виде	Матрица считана
4.	Чтение матрицы с клавиатуры	«вес» для матрицы	Матрица заполнена псевдослучайными числами
5.	Вывод матрицы как 3 массива	пункт меню	Матрица выведена на экран
6.	Вывод матрицы в обычном виде	пункт меню	Матрица выведена на экран
7.	Расчет суммы	две матрицы	Сумма матриц посчитана

8.	Вывод результата расчета	пункт меню	Посчитанная матрица-сумма выведена на экран
9.	Вывод меню	-	Повторный вывод меню в терминал
10.	Запрос теста сравнения алгоритмов	пункт меню	Результаты сравнения по памяти и времени

6. Оценка эффективности

Время (мкс): обычная эффективнее в n раз

Размер матрицы	10%			25%		
	Обычн. %	Разреж. %	Относит. эффективность	Обычн. %	Разреж. %	Относит. эффективность
5	2	14	700%	0	2	-
10	2	11	550%	1	3	300%
25	11	52	472%	3	17	567%
50	12	53	441%	7	54	771%
100	36	163	452%	91	254	279%

Память (байт): разреженная эффективнее в n раз

Размер матрицы	10%			25%		
	Обычн. %	Разреж. %	Относит. эффективность	Обычн. %	Разреж. %	Относит. эффективность
5	124	264	47%	124	264	47%
10	424	344	123%	424	424	100%
25	2524	1064	237%	2524	1624	155%
50	10024	3064	327%	10024	6424	156%
100	40024	9784	409%	40024	21784	184%

7. Вывод

Разреженные матрицы показывают бОльшую эффективность по памяти при меньшем проценте заполнения (200-300% против 150-180%). При любых значениях размера и % заполнения разреженные матрицы обрабатываются дольше обычных (около 300-400%). Для маленьких размеров матриц (<10) разреженные матрицы могут быть менее эффективны и по памяти (зависит от % заполнения).

8. Контрольные вопросы

1. Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?
   Разреженная матрица – матрица, содержащая большое кол-во нулей. Способы хранения: линейный связный список, кольцевой связный список, двунаправленные стеки и очереди, диагональная схема хранения симметричных матриц, связные схемы разреженного хранения.
2. Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?
   Для обычной матрицы выделяется m * n * sizeof(type) для хранения значений, и еще несколько десятков байт для хранения самого типа (размеры, указатели и т.д.)
   Для разреженной матрицы выделяется: k * sizeof(type) (k – кол-во ненулевых элементов) – 2 таких вектора; список - длиной равный кол-во строк.
3. Каков принцип обработки разреженной матрицы?

Обрабатываются и хранятся только ненулевые элементы, поэтому сокращается кол-во операций и затрачиваемой памяти тем сильнее, чем более матрица разрежена.

4. В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?
   Стандартные алгоритмы применяются для не разреженных матриц, а также когда скорость важнее занимаемой памяти. Разреженные матрицы тем эффективнее, чем сильнее они разрежены.