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

Студент **Паламарчук А.Н.**

Группа **ИУ7-33Б**

Предмет **Типы и структуры данных**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ **Паламарчук А.Н.**

Преподаватель _____ **Никульшина Т. А.**

Преподаватель _____ **Барышникова М. Ю.**

2023 г.

Условие задачи

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

Техническое 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;
- вектор JA, в элементе Nk которого находится номер компонент в A и IA, с которых начинается описание столбца Nk матрицы A.

1. Смоделировать операцию сложения двух матриц, хранящихся в этой форме, с получением результата в той же форме.
2. Произвести операцию сложения, 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.

Входные данные

- Команда (число) из меню программы (см. описание алгоритма) – обозначение необходимой операции.

Выходные данные

- Таблица эффективности
- Исходные матрицы
- Результирующая матрица

Возможные аварийные ситуации

- Некорректный исходный файл
- Ошибка открытия файла
- Отсутствие исходного файла
- Нехватка памяти для работы программы
- Некорректная команда
- Несовпадение количества строк и столбцов

Описание внутренних структур данных

```
int **m_1_classic = NULL;

int **m_2_classic = NULL;

typedef struct
{
    int *A;
    int *JA;
    int *IA;
} matrix_t;
```

Константы

```
#define COUNT_TESTS 100
```

Используемые функции

```
unsigned long long cur_mks_gettimeofday(void);
```

Возвращает текущее время в мкс

```
void free_matrix(int **matrix, size_t n);
```

Очищает матрицу

```
int fscanf_matrixs(char *src, int ***m_1_classic, int ***m_2_classic, int *row, int
*col, matrix_t *m_1, matrix_t *m_2, size_t *count_el, size_t print_flag);
```

Заполнение матриц из файла

```
int input_matrixs(int ***m_1_classic, int ***m_2_classic, int *row, int *col, matrix_t
*m_1, matrix_t *m_2, size_t *count_el);
```

Заполнение матриц вручную

```
void print_matrix(matrix_t m, size_t count_el, int col);
```

Распечатывает матрицу

```
int summary_matrix_classic(int **m_1_classic, int **m_2_classic, int row, int col,
size_t print_flag);
```

Классическое сложение матриц

```
int summary_matrix(matrix_t m_1, matrix_t m_2, size_t count_el, int col, size_t
print_flag);
```

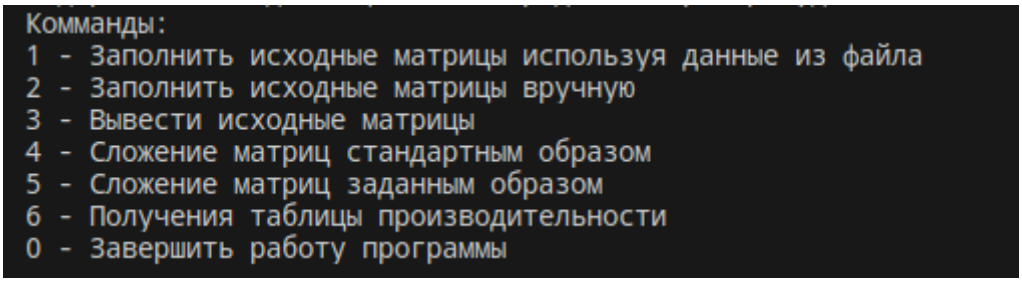
Сложение матриц в заданной форме

```
void efficency_table(void);
```

Вывод таблицы эффективности

Описание алгоритма

Программа обрабатывает нужную команду:



1. «Традиционное» сложение матрицы с матрицей. Проходим по всем элементам и попарно складываем значения элементов.

2. Сложение разреженных матриц

   2.1 Пустые столбцы исходной матрицы пропускаются

   2.2 Для не пустых столбцов определяется конечный индекс в массиве A

   2.3 Производится сложение ненулевых элементов, индексы строк определяются с помощью массива IA

   2.4 Сумма записывается в соответствующую ячейку результирующей матрицы

Производительность

Замеры проводились по 100 раз.

Время указано в мкс, затраты памяти в байтах
TIME

N	%	Regular matrix	Sparse matrix
100	10	50	5
100	20	64	7
100	30	68	11
100	40	88	18
100	50	85	21
100	60	91	31
100	70	94	35
100	80	91	40
100	90	88	51
100	100	85	58
200	10	228	14
200	20	265	24
200	30	300	44
200	40	338	65
200	50	358	88
200	60	377	116
200	70	380	126
200	80	373	152
200	90	351	164
200	100	332	177
300	10	483	28
300	20	562	51
300	30	634	91
300	40	696	129
300	50	762	180
300	60	822	231
300	70	840	262
300	80	823	296
300	90	769	316
300	100	726	349

MEMORY

N	%	Regular matrix	Sparse matrix
100	10	40000	8400
100	20	40000	16400
100	30	40000	24400
100	40	40000	32400
100	50	40000	40400
100	60	40000	48400
100	70	40000	56400
100	80	40000	64400
100	90	40000	72400
100	100	40000	80400
200	10	160000	32800
200	20	160000	64800
200	30	160000	96800
200	40	160000	128800
200	50	160000	160800
200	60	160000	192800
200	70	160000	224800
200	80	160000	256800
200	90	160000	288800
200	100	160000	320800
300	10	360000	73200
300	20	360000	145200
300	30	360000	217200
300	40	360000	289200
300	50	360000	361200
300	60	360000	433200
300	70	360000	505200
300	80	360000	577200
300	90	360000	649200
300	100	360000	721200

Выводы

Сложение разреженных матриц быстрее сложения обычных матриц, причём чем меньше процентное отношение количества ненулевых элементов к количеству всех элементов, тем больше разрыв между сложением разреженных матриц и сложением обычных матриц. Например, при 10% заполнении на 100 элементах время работы с разреженными составляет всего 10% от времени сложения обычных файлов. А, при 60% заполнении на 100 элементах время работы с разреженными составляет уже 33% от времени сложения обычных файлов. При > 50% заполнении матриц сложение разреженных начинает проигрывать по памяти стандартному алгоритму сложения.

Контрольные вопросы

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

   Разреженная матрица – это матрица с большим количеством нулевых элементов. Схемы хранения разреженных матриц: разреженный строчный формат, разреженный столбцовый формат, схема Кнута.

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

   Для хранения разреженной матрицы, выделяется память под хранение трех массивов, а количество ячеек памяти, можно рассчитать по формуле: 2*N + M, где N — количество ненулевых элементов, а M – количество столбцов.

   Для обычной матрицы выделяется N*M, N - количество строк, M - количество столбцов, ячеек памяти, при динамическом выделении памяти есть несколько способов выделить память под матрицу, способ выбирается программистом под конкретную задачу.

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

   При работе с разреженными матрицами алгоритм работает только с ненулевыми элементами матрицы, тогда количество операций будет пропорционально числу ненулевых элементов.

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

   Алгоритм работы с разреженными матрицами будет быстрее работать на достаточно больших матрицах с малым заполнением ненулевыми элементами. При небольших размерах матриц разумнее использовать традиционное сложение, это позволит упростить задачу понимания работы алгоритма. Кроме того, на достаточно больших матрицах, которые заполнены ненулевыми элементами, будет разумнее использовать традиционный алгоритм, т.к. такие матрицы не являются разреженными.