|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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\_\_\_\_\_\_Чыонг Ван Хао\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_*ИУ7И-31Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Вариант 24*

*Задача 2*

Принял : Барышникова М. Ю.

**Описание условия задачи**

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 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.

**Техническое задание**

**Входные данные**

Программа для получения данных из входного файла или вручную.

Select menu number:

1) Enter matrix A manually

2) Enter matrix B manually

3) Fill matrix A from file

4) Fill matrix B from file

5) Add matrices using the classical storage method

6) Add matrices using sparse columnar storage

7) Print original matrices in the classical form

8) Print original matrices in sparse columnar form

9) Print the resulting matrix in the classic form

10) Print the resulting matrix in sparse columnar form

11) Test result

0) Exit

1. Вручную введите матрицу A
2. Вручную введите матрицу B
3. Заполнить матрицу A из файла
4. Заполнить матрицу B из файла
5. Добавляем матрицы классическим способом хранения
6. Добавление матриц с использованием разреженного столбчатого хранилища
7. Распечатать оригинальные матрицы в классическом виде
8. Распечатать исходные матрицы в виде разреженных столбцов
9. Распечатайте получившуюся матрицу в классическом виде

10) Выведите полученную матрицу в виде разреженных столбцов

11) пройти тест

* Enter size matrix:

введите размер матрицы (квадратная матрица)

* Enter number of non\_zero:

введите количество ненулевых элементов

0) выйти из программы

**Выходные данные**

1. Исходные и результирующая матрицы в стандартном виде или разреженном столбцовом виде.

2. Сравните время выполнения и занятую память

**Функция программы**

Сложение матриц при классическом и разреженном столбцовом хранении.

**Обращение к программе**

Программа запускается из терминала командой «./app.exe» в директории с  
программой.

**Возможные аварийные ситуации и ошибки пользователя**

1. Ввести неправильный выбор.

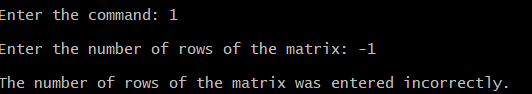
На выходе сообщение: «ERROR COMMAND.»



2. Неправильно введите количество строк матрицы (Вручную).

На выходе сообщение:

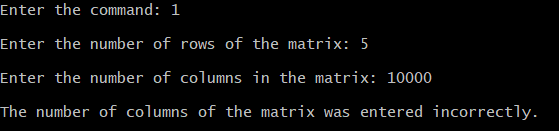
« The number of rows of the matrix was entered incorrectly. »



3. Неправильно введите количество столбцов матрицы

На выходе сообщение:

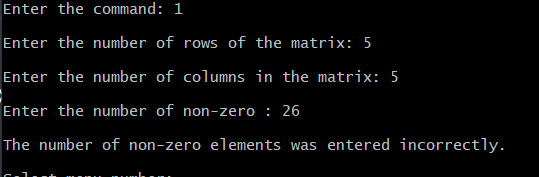
« The number of columns of the matrix was entered incorrectly. »



4. Ввод количества ненулевых элементов матрицы неверен

На выходе сообщение:

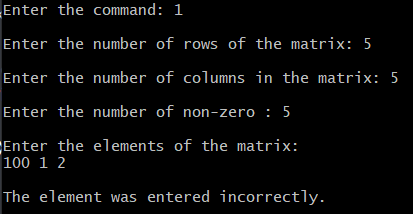
« The number of non-zero elements was entered incorrectly. »



5. Неправильный ввод данных элемента матрицы

На выходе сообщение:

« The element was entered incorrectly. »



6. Введите неправильное данные из файла.

На выходе сообщение: «ERROR INPUT!»

7. Параметр функции передан с ошибкой

На выходе сообщение: «ERROR PARAMETER.»

8. Ошибка выделения памяти

На выходе сообщение: «ERROR ALLOCATE.»

9. Выполнить сложение матриц, которые не существуют

На выходе сообщение: «Matrix A not exist.»

или «Matrix B not exist.»

10. Размер двух матриц не совпадает

На выходе сообщение: «Wrong size of matrix!»

**Внутренняя структура данных**

Для хранения матрицы использую структуры:

стандартная форма:

typedef struct

{

    int \*\*matrix;

    int rows;

    int cols;

} *matrix*;

разреженная матричная форма:

typedef struct

{

    int \*num;

    int \*rows;

    int \*cols;

    int non\_zero;

} *sparse\_matrix*;

**Функции программы**

void conditions(void);

Описание: распечатать условие входных данных на экран.

void menu(void);

Описание: печать на экран параметры выполнения.

void free\_matrix(*matrix* \**mtr*, int *n*);

Описание: освобождение памяти классической матрицы.

void free\_sparse(*sparse\_matrix* \**mtr*);

Описание: освободить память разреженной матрицы.

void free\_matrix\_only(int \*\**matrix*, int *n*);

Описание: освободить память только матриц (не структур).

int create\_matrix(*matrix* \**mtr*);

Описание: получить данные о размере с клавиатуры и выделить память для матрицы.

int input\_matrix\_by\_hand(*matrix* \**mtr*, *sparse\_matrix* \**sparse*);

Описание: получить данные элементов в матрице вручную.

int \*\*file\_input(int \**rows*,int \**cols*, int \**count*, *FILE* \**file*);

Описание: получить элементы данных в матрице из файла.

Входные значения: входной файл

Выходные значения: код ошибки и сообщение (если есть).

void random\_matrix(*matrix* \**mtr*, const int *size*);

Описание: выполнить заполнение, как сгенерировать значения элементов матрицы случайным образом.

int print\_matrix(*matrix* *mtr*);

Описание: вывод на экран матрицы в классическом виде.

int print\_sparse(*sparse\_matrix* *matrix*, const int *count*);

Описание: выводить на экран матрицу в разреженном виде.

int sparse\_convert(*matrix* *mtr*, *sparse\_matrix* \**sparse*);

Описание: преобразовать матрицу из классической в ​​разреженную форму.

int classic\_add(*matrix* *matrix\_A*, *matrix* *matrix\_B*, *matrix* \**result*);

Описание: выполнить сложение матриц классическим способом.

int count\_result\_non\_zero(*sparse\_matrix* *sparse\_a*, *sparse\_matrix* *sparse\_b*, const int *column*, int \**result\_count*);

Описание: подсчитать количество ненулевых элементов в результирующей разреженной матрице.

int sparse\_sum(*sparse\_matrix* *sparse\_a*, *sparse\_matrix* *sparse\_b*, *sparse\_matrix* \**result*, const int *column*);

Описание: Добавление матриц с использованием разреженного столбчатого хранилища.

int create(*matrix* \**mtr*, int const *size*);

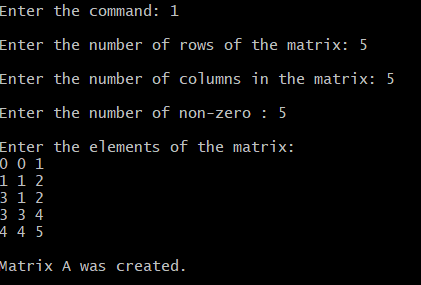
Описание: выделение памяти для квадратной матрицы.

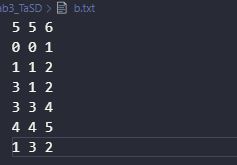
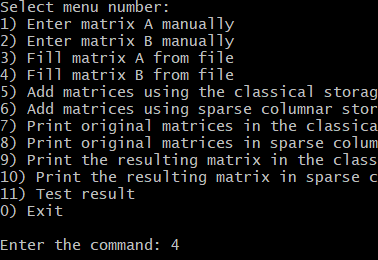
void test\_time();

Описание: выполнять тесты на время безотказной работы и использование памяти методами сложения.

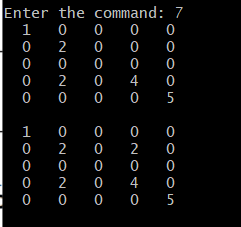
**Тесты**

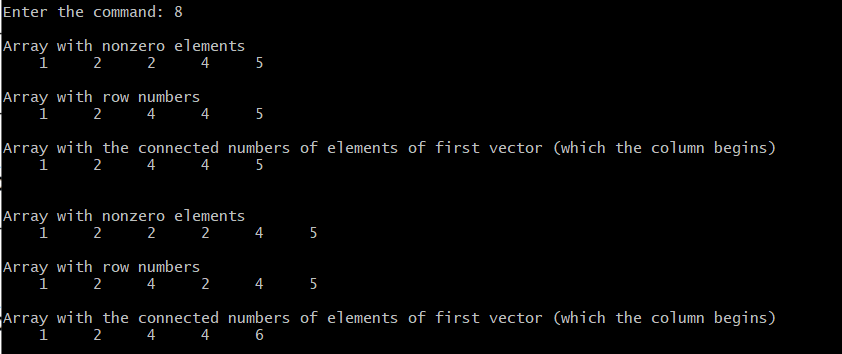
Ввод значений (вручную и считать из файла)





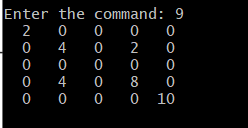
Показать исходную матрицу





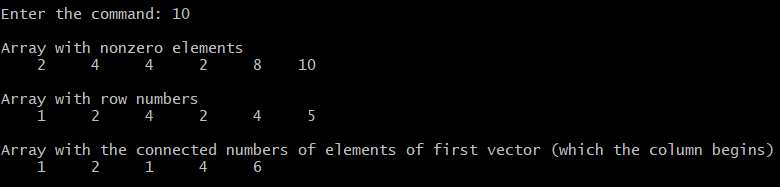
Добавляем матрицы классическим способом хранения





Добавление матриц с использованием разреженного столбчатого хранилища





**Оценка эффективности**

Время сложения (микросекунды)

|  |  |  |  |
| --- | --- | --- | --- |
| Размер | количество ненулевых чисел | Обычная матрица | Разреженная матрица |
| 10x10 | 20 | 2 | 6 |
| 100x100 | 200 | 43 | 27 |
| 200x200 | 400 | 259 | 86 |
| 500x500 | 1000 | 1634 | 124 |
| 1000x1000 | 2000 | 3555 | 250 |
| 10x10 | 100 | 2 | 5 |

Объем занимаемой памяти (байты) :

|  |  |  |  |
| --- | --- | --- | --- |
| Размер | количество ненулевых чисел | Обычная матрица | Разреженная матрица |
| 10x10 | 20 | 496 | 448 |
| 100x100 | 200 | 40816 | 4780 |
| 200x200 | 400 | 161616 | 9580 |
| 500x500 | 1000 | 1004016 | 23980 |
| 1000x1000 | 2000 | 4008016 | 47992 |
| 100x100 | 100 | 496 | 1228 |

**Выводы**

Добавление двух матриц в разреженной форме занимает меньше времени, чем в нормальной форме. По мере увеличения размера матрицы временная эффективность метода добавления двух разреженных матриц становится более очевидной.

Эффективность по памяти:

Работа с разреженной матрицей займет меньше памяти, чем работа с обычной матрицей, если матрица заполнена не полностью.

**Контрольные вопросы**

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

Разреженная матрица – это матрица, содержащая большое количество нулей. Способы хранения: связная схема хранения, строчный формат, столбцовый формат, линейный связный список, кольцевой связный список.

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

Под обычную матрицу выделяет N \* M ячеек памяти, где N – строки, а M – столбцы. Для разреженной матрицы – зависит от способа. В случае разреженного столбцового формата, требуется 2 \* Q + M ячеек памяти, где Q – количество ненулевых элементов, M — число столбцов.

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

Алгоритмы обработки разреженных матриц предусматривают действие только с ненулевыми элементами, и, таким образом, количество операций будет пропорционально количеству ненулевых элементов.

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

Стандартные алгоритмы обработки матриц эффективнее применять при большом количестве ненулевых элементов. Cтоит отметить, что если не так важна память, занимаемая матрицами, но важно время, то в случае сложения лучше так же воспользоваться стандартными алгоритмами сложения матриц.