|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 3

По дисциплине «Типы и структуре данных»

### Название Обработка разряженных матриц

### Студент Блохин Артем Олегович

*фамилия, имя, отчество*

### Группа ИУ7-34Б

Тип лабораторной работы: Учебная

### Название

предприятия

|  |  |  |
| --- | --- | --- |
| Студент |  | Блохин А.О. |
| Преподаватель | *подпись, дата* | *фамилия, и.о.*  Силантьева А.В |
|  | *подпись, дата* | *фамилия, и.о.* |

*2022 г.*

***Содержание***

### Условие задачи

### Техническое задание

### Описание алгоритма

### Анализ алгоритмов

### Контрольные вопросы

### Заключение

***Цель работы -*** реализация алгоритмов обработки разреженных матриц, сравнение этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

***Условия задачи***

***Вариант (14 % 6) + 1 = 3***

*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****.*

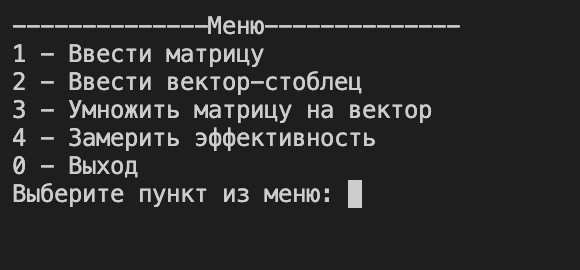
*1. Смоделировать операцию умножения матрицы и вектора-столбца, хранящихся в этой форме, с получением результата в той же форме.*

*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.*

***Требование к работе с программой:***

- Взаимодействие с программой строго по меню.



- Водить можно любые цифры за значение меню, в случае не нахождения такого пункта меню выводит сообщение об ошибке.

- Чтобы выйти из программы необходимо вести «0».

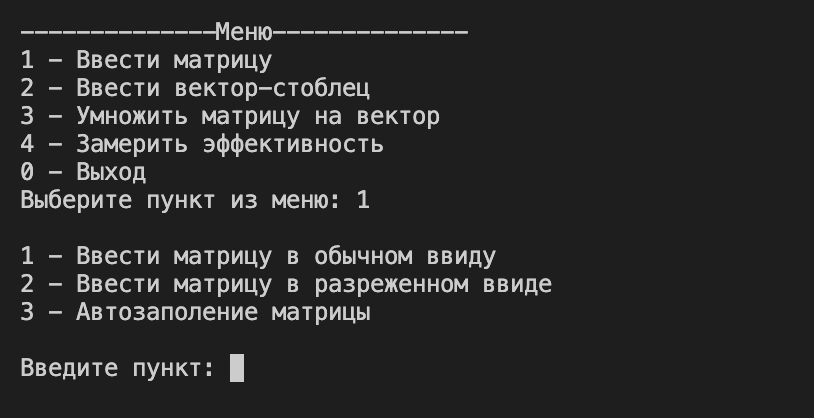
- При вводе существующих пунктов пользователю предоставляются данные, либо выводится подменю, либо выходит

- Взаимодействие в пунктах 1 и 2 по подменю:

- Чтобы вести матрицу или вектор в любом ввиде, заполняются все поля вводя правильно (при некорректных слуаях выводится сообщение об ошибке).

- Заданы ограничения ввода при работе с матрицами, то есть размер 5000 x 5000.

- Вывод обычной матрицы осуществляется до размерности столбцов или строк 30 элементов



***Описание алгоритма:***

В реализации программы были созданы две структуры для хранения обычной и разряженной матрицы:

*Обычная матрица*

typedef struct   
{  
 size\_t n; // количество строк  
 size\_t m; // количество столбцов

int \*\*matrix; // указатель на матрицу  
} usual\_matrix\_t;

*Разреженная матрица*

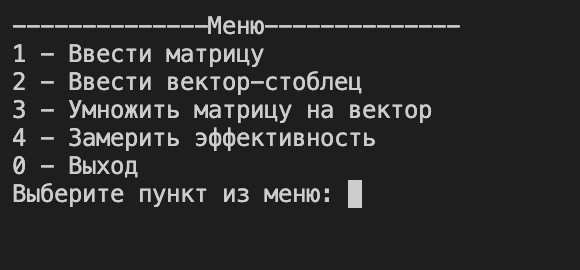
typedef struct st\_list  
{  
 int \*list; // указатель на массив для связного списка  
} list\_t;

typedef struct  
{  
 int n; // количество строк  
 int m; // количество столбцов  
 int na; // количество ненулевых элементов  
  
 int \*A; // указатель на массив с ненулевыми элементами  
 int \*JA; // указатель на массив с индексами j ненулевых элементов  
 list\_t IA; // связный список  
} special\_matrix\_t;

*Разреженная вектора столбца*

typedef struct  
{  
 int n; // количество элементов вектора  
 int na; // количество ненулевых элементов вектора  
  
 int \*A; // массив ненулевых элементов  
 list\_t IA; //связный список  
} special\_vector\_column\_t ;

Для взаимодействия с программой было создано консольное меню:

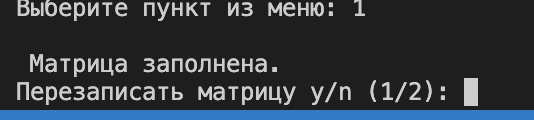


При вводе корректного значения выполняются определенные действия или операции, при некорректном выводится в консоль сообщение об ошибке. Так происходит пока не будет введен корректное значение меню.

Для реализации программы, была использована динамическая типизация, поэтому были написана функции выделения памяти и очищения памяти, чтобы при работе не возникали утечки памяти.

*Пункта 1 консольного меню:*

При вводе матрицы учитывается, что пользователь мог ввести ее заранее, поэтому у него спрашивается необходимо ли перезаписать ее.

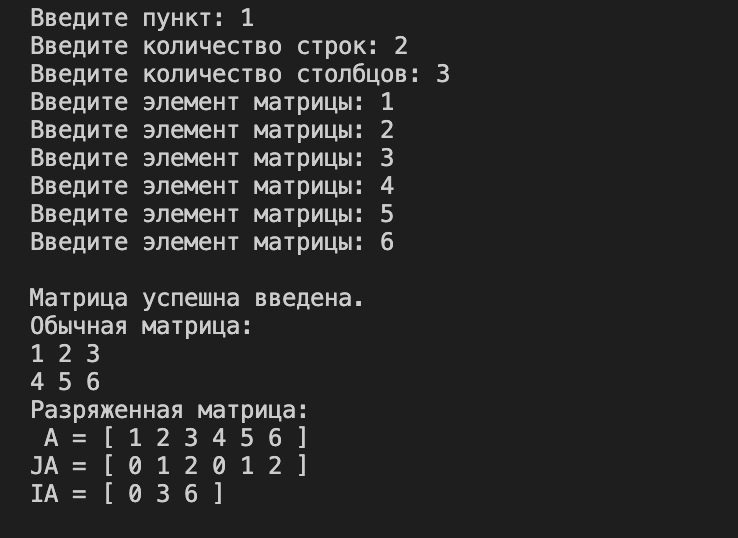


В качестве ответа да принимается «1», а нет – «2».

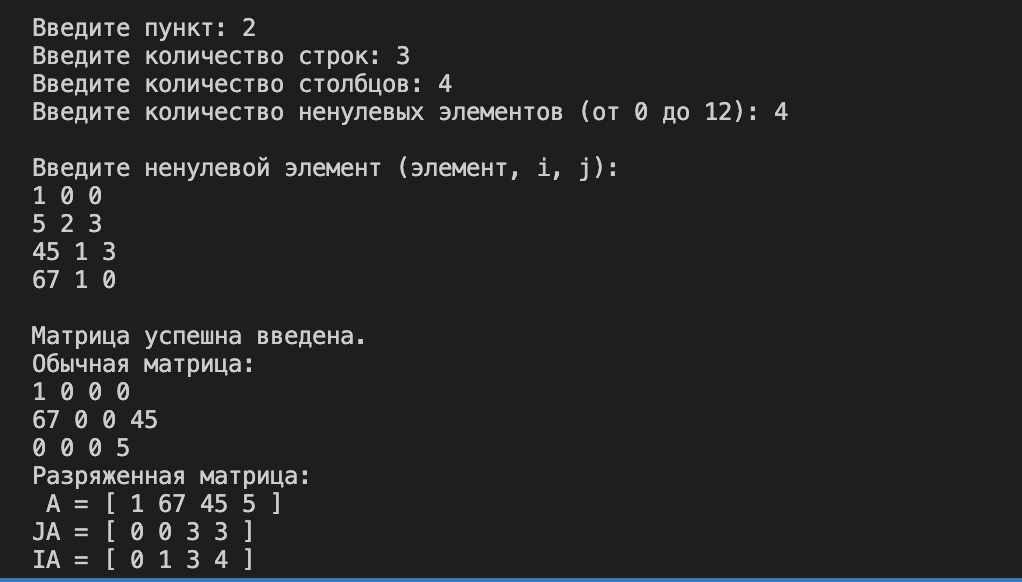
При вводе ответа «1»: очищается данные записанные в эту матрицу, затем очищается память, ранее выделенная, а уже потом проводится обычная ситуация ввода, как при первом вводе матрицы.

При вводе пользователь выбирает каким образом он будет вводить матрицу:

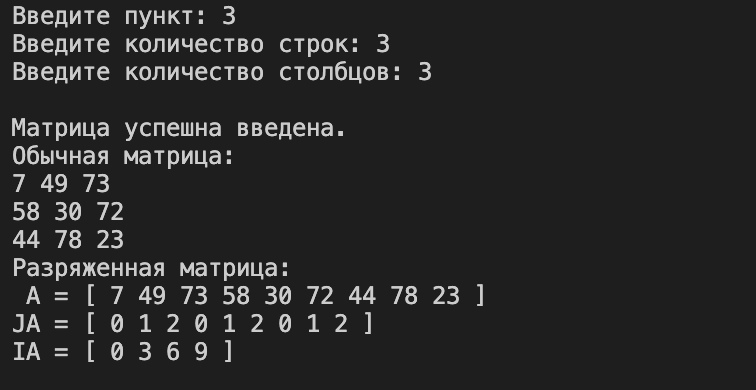
1. Обычный ввод, реализован простой ввод матрицы, то есть ввод размеров и ввод каждого элемента отдельно:



1. Разреженный ввод матрицы, означает, что необходимо ввести размеры матрицы, количество ненулевых элементов и сами ненулевые элементы, в виде «элемент номер строки номер столбца», если введен существующий элемент или же индекс неверный, то пользователю сообщается об этом и он заново вводит элемент (пока не ведет все элементы матрицы):



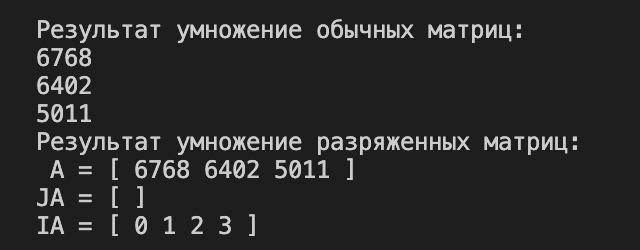
1. Автозаполнение матрицы, означает, что нужно ввести размеры матрицы и также процент ее заполнения:



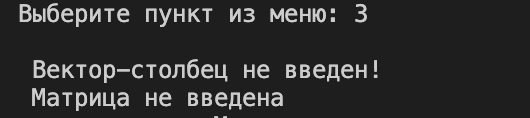
*Пункт 2 консольного меню* реализован аналогично первому, только здесь ввод осуществляется вектора-столбца, то есть матрицы «n x 1».

*Пункт 3 консольного меню* осуществляет умножение двух видов:

* В обычном алгоритм умножения матриц
* Умножение разряженных матриц



- При вводе значение этого пункта меню, и если матрица или вектор для расчетов не введены, то выводится сообщение об том, что именно пользователь не ввел



*Пункт 4 консольного меню* осуществляет замеры времени работы и памяти в байтах. (при этом производится заполнение матриц от 0 – 100 % с шагом 10%). Матрицы создаются, как в обычном в виде, так и в разреженном, затем производится замеры времени алгоритмов умножения и память, которую занимают матрица и два вектора-столбца. Все полученные данные выводятся в виде таблиц, рассчитывается эффективность разреженного метода в процентах, а также выводится процент заполнения и количество ненулевых элементов в матрицах и в векторе-столбце (для выполнения умножения).

*Пункт 0 консольного меню* Выход из программы и очищение памяти матрицы и двух векторов-столбцов

Заполнение обычной матрицы осуществляется по вводу всех элементов подряд, разреженный – только ненулевые элементы, при этом сначала создается обычная матрица в программе, затем уже разреженный вид матрицы (в связном списке нулевые строки заполняются значением индексами прошлых элементов массива ненулевых элементов).

Умножение разреженных матриц происходит следующим образом:

В начале в связном списке IA просматривается два первых элемента и если они отличаются друг от друга, то есть второй элемент больше первого, что означает, что в строке есть ненулевые элементы, при этом разница между 1 и 2 элементов – количество ненулевых элементов в строке. Затем из массива номеров столбцов JA просматривается индекс столбца нужного ненулевого элемента, после чего по этому индексу столбца смотрится существует ли ненулевой элемент в векторе-столбце такой индекс строки, если да, то происходит умножение элементов. Общий результат умножения строки матрицы на вектор-столбец, отличного от нуля записывается в определенным индексом строки в новый вектор-столбец.

***Аварийные выходы из программы***

Выходов аварийных нет, в том случае, если программа не будет принудительна закрыта.

Выход осуществляется только пунктом меню «0»

В ошибочных случаях выводится сообщение об ошибке и ввод повторяется, пока программа не получит корректные данные.

Если в результате операции – умножения или вводе матрицы и векторов, память не была выделена, то пользователь возвращается в главное меню и выводится сообщение об не выделении память в определенной операции.

***Анализ эффективности методов матриц***

Для замеров эффективности хранение и времени операции умножения разреженного метода были взяты несколько размеров матрицы **(n x m)** и векторов-столбецов**(m)**:

* Матрица 150 x 150 и вектор-столбец 150 элементов
* Матрица 600 х 500 и вектор-столбец 500 элементов
* Матрица 1200 х 1300 и вектор-столбец 1300 элеметов

Хранение матрицы в обычном ввиде измеряется - **n \* m \* sizeof(int)**,а для вектора-столбца в обычном ввиде **n \* sizeof(int).**

Хранение матрицы в разряженном ввиде измеряется – **2 \* na \* sizeof(int) + (n + 1) \*****sizeof(int)**, а для вектора-столбца - **na \* sizeof(int) + (n + 1) \* sizeof(int)** ,

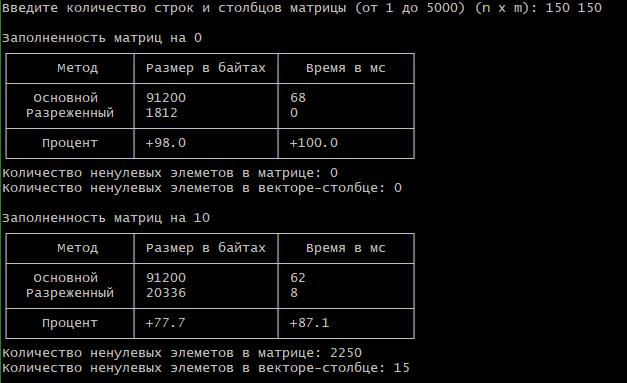
где:

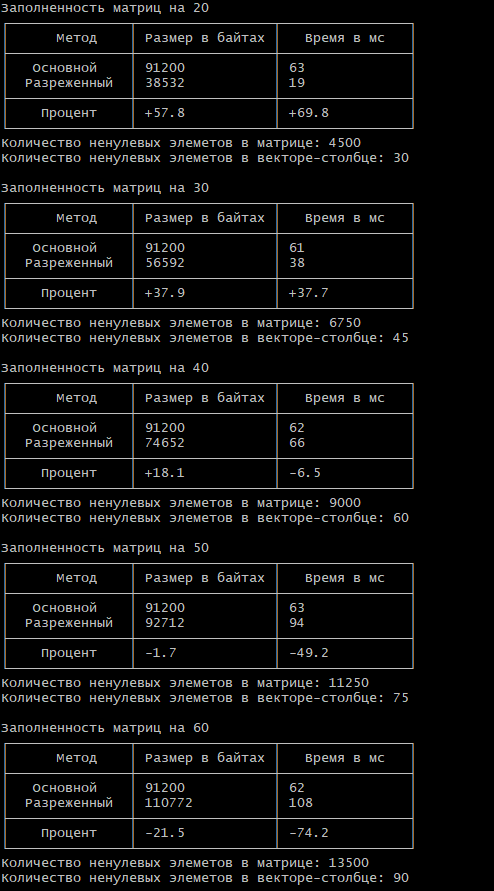
* **n**  - количество столбцов
* **m** – количество строк
* **na** – количество ненулевых элементов
* **sizeof(int)** – размер целого числа в байтах

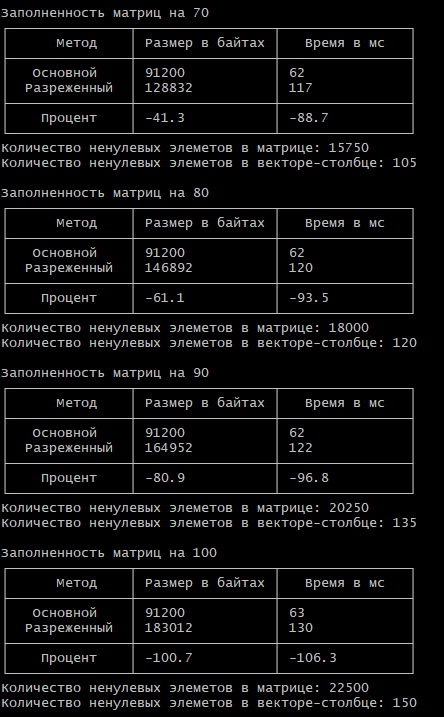
В таблицу выводится сумма хранение матрицы и двух векторов-столбцов.

Время выполнении операции умножение вычисляется выполнение этой операции несколько раз

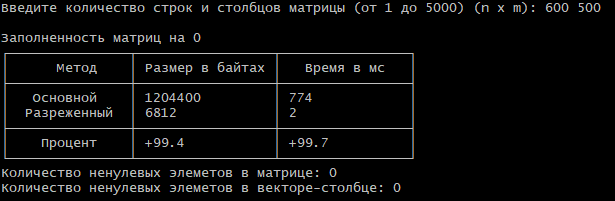
* ***Замеры матрица на 150 х 150 и вектор-столбец 150***

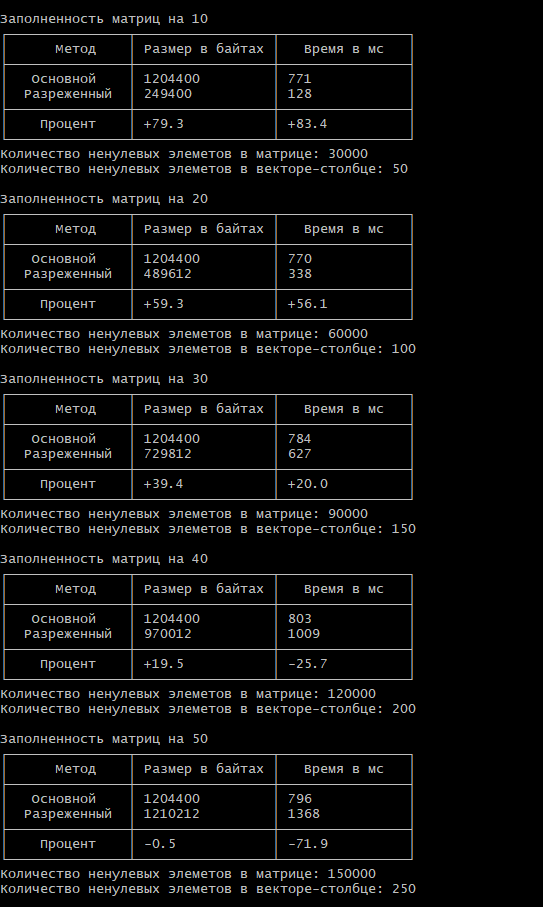


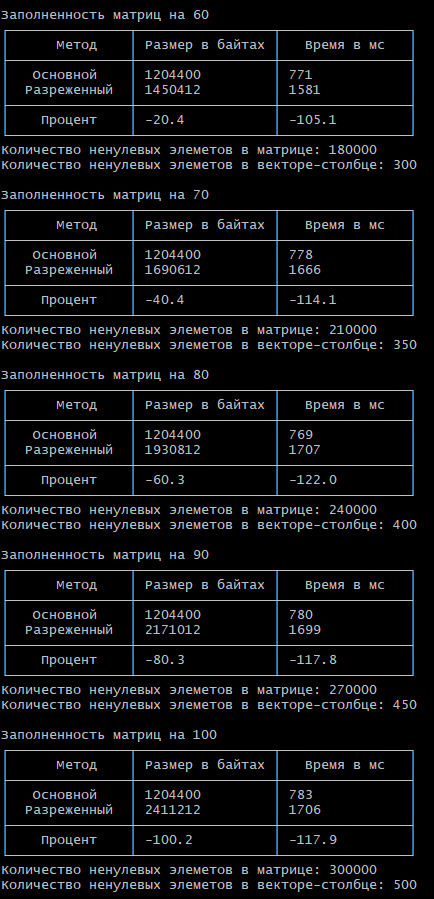




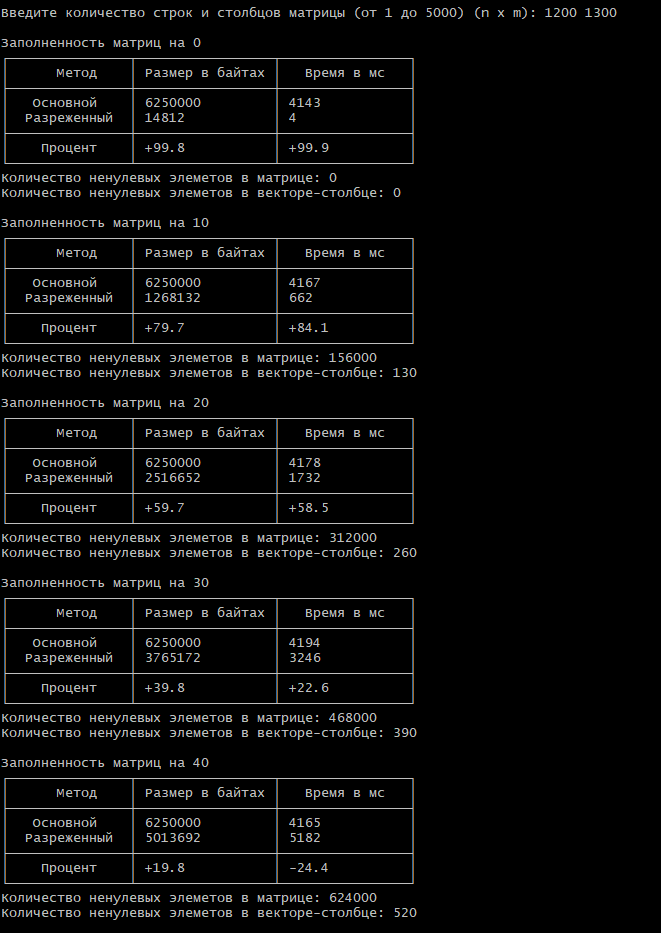
* ***Замеры матрица на 600 х 500 и вектор-столбец 500***

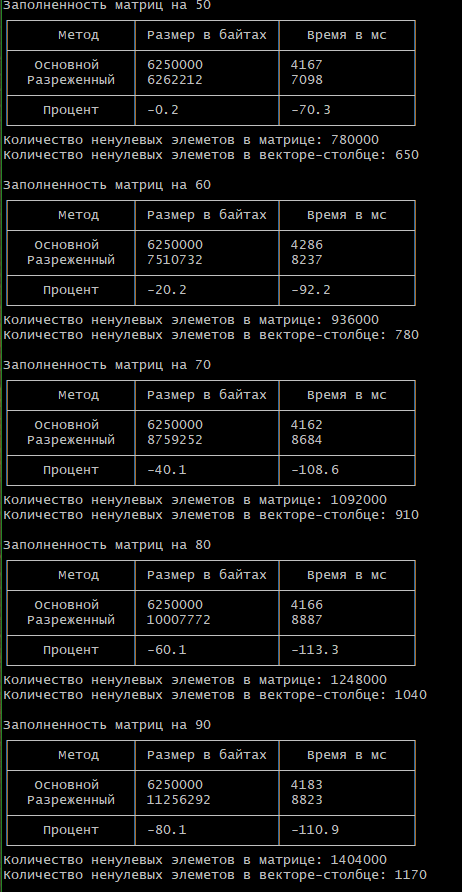


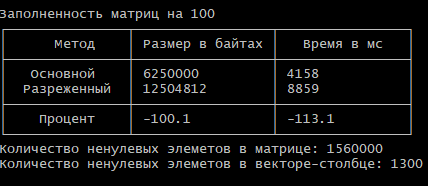




* ***Замеры матрица на 1200 х 1300 и вектор-столбец 1300***







В результате измерений, были получены выше предоставленные данные, по которым можно сделать вывод, что каждый из 2-х методов эффективный в разных ситуациях, так:

* *Разреженный метод*, эффективнее использовать по времени выполнения операции (в данном случае умножение) при разреженности матрицы более, чем 60% (40% заполненности).  
  Хранение же матриц и векторов-столбцов эффективнее хранить, когда нулевых элементов матрицы больше, чем ненулевых элементов. По результатам такая эффективность достигает при разреженности матрицы и векторов-столбцов больше, чем 50%. (50% заполненности).
* *Простой метод*, эффективнее использовать при заполненности матрицы больше, чем 70% ненулевыми элементами, как по времени выполнения операции, так по хранению матриц и векторов-столбцов.

***Контрольные вопросы***

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

Разреженная матрица – матрица с преимущественно нулевыми элементами. Число ненулевых элементов в матрице порядка n может выражаться как **n^(1+g),** где **g < 1.** Значения g лежат в интервале 0.2 ... 0.5.

Существуют различные методы хранения элементов матрицы в памяти.

Например, линейный связный список, т.е. последовательность ячеек, связанных в определенном порядке. Каждая ячейка списка содержит элемент списка и указатель на положение следующей ячейки.

Можно хранить матрицу, используя кольцевой связный список, двунаправленные стеки и очереди.

Существует диагональная схема хранения симметричных матриц, а также связные схемы разреженного хранения.

Связная схема хранения матриц, предложенная Кнутом, предлагает хранить в массиве (например, в AN) в произвольном порядке сами элементы, индексы строк и столбцов соответствующих элементов (например, в массивах I и J), номер (из массива AN) следующего ненулевого элемента, расположенного в матрице по строке (NR) и по столбцу (NC), а также номера элементов, с которых начинается строка (указатели для входа в строку – JR) и номера элементов, с которых начинается столбец (указатели для входа в столбец - JC).

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

Для хранения обычной матрицы: **N \* M \* sizeof(elem).** Память под разреженную матрицу (**K \* sizeof(elem) + (N + 1) \* sizeof(elem)**) выделяется в зависимости от схемы хранения. Память выделяется по мере наполнения ненулевыми элементами.

Пояснение:

* N – количество строк
* M – количество столбцов
* K – количество ненулевых элементов
* elem – тип данных

*3. Каков принцип обработки разреженной матрицы?*

Обработка разреженной матрицы предполагает работу только с ненулевыми элементами (таким образом, количество операций пропорционально количеству ненулевых элементов).

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

Разреженность матрицы следует учитывать только в том случае, если из этого можно извлечь выгоду за счёт игнорирования нулевых элементов.

При достижении определенного процента наполнения ненулевыми элементами происходит значительное падение эффективности по времени.

***Вывод***

В результате выполнения данной лабораторной работы, было использованы два алгоритма хранения и операции над матрицами, пори этом использование разряженного способа матрицы оправдывает свою эффективность при небольшом заполнении матрицы ненулевыми элементами, что позволяет более эффективно обращаться с памятью и временем выполнения программы. Однако есть случае, когда матрица заполнена ненулевыми элементами больше, чем нулевыми, то эффективность способа падает, как по памяти, так и по времени выполнения программы.

Стандартный же алгоритм эффективнее использовать при тех случаях, когда матрица заполнена больше 70% ненулевых элементов. В других случаях разреженный метод работает быстрее по времени, но по памяти в случае заполнение матрицы более чем на 50% ненулевыми элементами, он занимает больше памяти, чем стандартный.