1. Описание условия задачи.

Первая 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.

Вторая разреженная матрица хранится в форме 3-х объектов:

- вектор B содержит значения ненулевых элементов;

- вектор JB содержит номера столбцов для элементов вектора A;

- вектор IB, в элементе Nk которого находится номер компонент в B и JB, с которых начинается описание строки Nk матрицы B.

1. Смоделировать операцию умножения двух матриц, хранящихся в указанной форме, с получением результата в форме хранения первой матрицы.

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.

2. Техническое задание.

Исходные данные и правила их ввода.

- Пункт меню: числа от 1 до 14.
- Матрица:
    - Элемент матрицы – целое число от -2^{63} до $2^{63}-1$.
    - Размер матрицы – целое положительное число больше 1. Ограничен типом size_t.
    - Количество ненулевых – целое положительное число, не превышающее количество элементов матрицы.
    - Позиция в матрице – от 1 до размера матрицы включительно.
    - Процент заполнения – положительное вещественное число, не превышающее 100.
- Матрица-результат умножения двух матриц:
    - В случае ошибки на экран выводится сообщение об ошибке в формате «ОШИБКА: описание ошибки».
    - Матрица в csc-формате.
    - Если количество столбцов и строк матрицы не превышает 50 – Матрица в обычном формате.

Способ обращения к программе:

Запуск исполняемого файла. В рабочей директории выполнить команду ./app.exe.

Аварийные ситуации:

- Ввод некорректных данных.
- Ошибка выделения памяти.
- Несовпадение количества столбцов первой матрицы и количество строк второй.

В случае аварийной ситуации выводится сообщение об ошибке.

3. Описание внутренних структур данных.

Структура обычной матрицы:

```
typedef struct {
    int64_t **rows;
    size_t rows_count;
    size_t columns_count;
} matrix_t;
```

Структура содержит 3 поля:

- `rows` – массив указателей на строки матрицы – указатель на указатель типа `int64_t`.
- `rows_count` – количество строк матрицы – беззнаковое целое числа типа `size_t`.
- columns_count – количество столбцов матрицы - беззнаковое целое числа типа `size_t`.

Структура целочисленного массива:

```
typedef struct int_array{
    int64_t *data;
    size_t size;
} int_array_t;
```

Структура содержит 2 поля:

- `data` – массив целых чисел типа `int64_t`.
- `size` – размер массива - беззнаковое целое числа типа `size_t`.

Структура массива чисел типа size_t.

```
typedef struct size_array{
    size_t *data;
    size_t size;
} size_array_t;
```

Структура содержит 2 поля:

- `data` – массив беззнаковых целых чисел типа `size_t`.
- `size` – размер массива - беззнаковое целое числа типа `size_t`.

Структура csc-матрицы (Матрица по столбцам):

```
typedef struct {
    int_array_t *data;
    size_array_t *indices;
    size_array_t *pointers;

    size_t original_rows_count;
} csc_matrix_t;
```

Структура содержит 4 поля:

- `data` – указатель на массив ненулевых элементов – массив целых чисел типа `int_array_t`.
- `indices` - указатель на массив номеров строк элементов типа – массив беззнаковых целых типа `size_array_t`.
- `pointers` - указатель на массив номеров элементов, с которых начинается описание очередного столбца в массивах `data` и `indices` – массив беззнаковых целых типа `size_array_t`.
- `original_rows_count` - количество строк матрицы – беззнаковое целое типа `size_t`.

Структура csr-матрицы (Матрица по строкам):

```
typedef struct {
    int_array_t *data;
    size_array_t *indices;
    size_array_t *pointers;

    size_t original_columns_count;
} csr_matrix_t;
```

Структура содержит 4 поля:

- `data` – указатель на массив ненулевых элементов – массив целых чисел типа `int_array_t`.
- `indices` - указатель на массив номеров строк элементов типа – массив беззнаковых целых типа `size_array_t`.
- `pointers` - указатель на массив номеров элементов, с которых начинается описание очередного столбца в массивах `data` и `indices` – массив беззнаковых целых типа `size_array_t`.
- `original_columns_count` - количество столбцов матрицы – беззнаковое целое типа `size_t`.

Структура данных исследования:

```
typedef struct {
    size_t size;
    double filling;
    unsigned int tests;
    double average_time_matrix_mult_matrix;
    size_t average_size_matrix_mult_matrix;
    double average_time_csc_mult_csr;
    size_t average_size_csc_mult_csr;
} research_t;
```

Структура содержит 7 полей:

- `size` - размер квадратных матриц, над умножением которых производится исследование – беззнаковое целое число типа `size_t`.
- `filling` - процент заполнения матриц – вещественное число типа `double`.
- `tests` - количество произведённых измерений – беззнаковое целое число типа `unsigned int`.
- - среднее время умножения матрицы на матрицу методом строка на столбец – вещественное число типа `double`.
- - средний размер, занимаемый обычными матрицами – беззнаковое целое число типа `size_t`.
- - среднее время умножения csc-матрицы на csr-матрицу – вещественное число типа `double`.
- - средний размер, занимаемый csc и csr матрицами – беззнаковое целое число типа `size_t`.

Для работы со структурами используются следующие функции:

`csc_matrix_t *csc_matrix_from_matrix(matrix_t *matrix)` – создание csc-матрицы из обычной.

`void csc_matrix_delete(csc_matrix_t *csc_matrix)` – освобождение памяти, выделенной под csc-матрицу.

`matrix_t *matrix_from_csc_matrix(csc_matrix_t *csc_matrix)` – создание обычной матрицы по csc-матрице.

`void csc_matrix_print(csc_matrix_t *csc_matrix)` – печать csc-матрицы на экран в формате трёх массивов.

`csr_matrix_t *csr_matrix_from_matrix(matrix_t *matrix)` – создание csr-матрицы из обычной.

`void csr_matrix_delete(csr_matrix_t *csr_matrix)` – освобождение памяти, выделенной под csr-матрицу.

`matrix_t *matrix_from_csr_matrix(csr_matrix_t *csr_matrix)` – создание обычной матрицы по csr-матрице.

`void csr_matrix_print(csr_matrix_t *csr_matrix)` – печать csr-матрицы на экран в формате трёх массивов.

`matrix_t *matrix_mult_matrix(matrix_t *matrix_1, matrix_t *matrix_2)` – умножение обычных матриц.

`csc_matrix_t *csc_multiply_csr(csc_matrix_t *csc_matrix, csr_matrix_t *csr_matrix)` – умножение csc-матрицы на csc-матрицу.

4. Описание алгоритма.

Выводится меню программы. Пользователь вводит номер команды, после чего выполняется действие.

Алгоритм умножения:

На вход подается csc-матрица и csr-матрица. Если количество столбцов первой матрицы не совпадает с количеством строк второй, происходит выход из программы. Если количество ненулевых элементов одной из матриц равно нулю, создается и возвращается пустая матрица-результат.

Исходная csc-матрица (1 матрица) транспонируется. Это нужно для более быстрого обхода матрицы по строкам.

Затем csr-матрица (2 матрица) переводится в csc формат: её данные копируются в структуру csc-матрицы (при таком копировании данных полученная транспонированная матрица в csc формате), а затем полученная csc-матрица транспонируется. Таким образом организуется более быстрый обход матрицы по столбцам.

Далее создается матрица-результат в csc-формате. Изначально размер внутренних векторов матрицы способен вместить полностью заполненную матрицу.

Создается переменная-счетчик количества записанных в матрицу ненулевых элементов и их индексов, а также количества записанных указателей.

Программа при помощи цикла проходит по столбцам 2 матрицы. Для каждого столбца 2 матрицы программа при помощи цикла проходит по строкам 1 матрицы. Создается переменная суммы результатов умножений для данной строки матрицы 1, равная 0. Далее, для каждого ненулевого элемента данного столбца 2 матрицы, программа ищет соответствующий ненулевой элемент в строке 1 матрицы при помощи векторов индексов. При совпадении индексов к переменной суммы прибавляется результат умножения элементов. При завершении обхода очередной строки матрицы 1 значение переменной суммы дописывается к вектору ненулевых элементов матрицы-результата, так же к вектору индексов матрицы-результата дописывается данный номер строки 1 матрицы. После завершения обхода очередного столбца матрицы 2 количество записанных в матрицу-результат ненулевых элементов добавляется к вектору указателей матрицы-результата.

По завершении умножения, вектора матрицы-результата сокращаются до нужного значения, после чего возвращается матрица-результат.

5. Оценка эффективности.

При проведении исследования квадратные матрицы размером от 100 до 500 шагом в 100 при проценте заполнения от 0 до 66 умножаются друг на друга по 10 раз при помощи обычного алгоритма умножения и при помощи умножения csc-матрицы на csr-матрицу.

Время умножения матриц.

Размер	Заполнение	Обычные матрицы, тики	Разреженные матрицы, тики
100	0,00%	1,57E+07	2,75E+03
100	1,00%	1,59E+07	1,12E+06
100	5,00%	1,57E+07	6,83E+06
100	8,00%	1,58E+07	1,44E+07
100	9,00%	1,55E+07	1,77E+07
100	46,00%	1,53E+07	2,85E+08
100	56,00%	1,53E+07	3,94E+08
200	0,00%	1,36E+08	3,24E+03
200	1,00%	1,36E+08	7,50E+06
200	5,00%	1,37E+08	8,33E+07
200	8,00%	1,36E+08	1,88E+08
200	9,00%	1,37E+08	2,32E+08
200	46,00%	1,36E+08	4,32E+09
200	56,00%	1,36E+08	5,96E+09
300	0,00%	4,79E+08	4,89E+03
300	1,00%	4,80E+08	2,86E+07
300	5,00%	4,79E+08	3,78E+08
300	8,00%	4,80E+08	8,87E+08
300	9,00%	4,79E+08	1,10E+09
300	46,00%	4,79E+08	2,15E+10
300	56,00%	4,79E+08	2,96E+10
400	0,00%	1,18E+09	8,11E+03
400	1,00%	1,18E+09	7,80E+07
400	5,00%	1,18E+09	1,14E+09
400	8,00%	1,18E+09	2,71E+09
400	9,00%	1,18E+09	3,40E+09
400	46,00%	1,18E+09	6,85E+10
400	56,00%	1,18E+09	9,33E+10
500	0,00%	2,46E+09	7,57E+03
500	1,00%	2,46E+09	1,70E+08
500	5,00%	2,46E+09	2,68E+09
500	8,00%	2,46E+09	6,55E+09
500	9,00%	2,45E+09	8,20E+09
500	46,00%	2,45E+09	1,64E+11
500	56,00%	2,45E+09	1,97E+11

Размер, требуемый для хранения матриц.

Размер	Заполнение	Обычные матрицы, байты	Разреженные матрицы, байты
100	0,00%	1,60E+05	1,62E+03
100	1,00%	1,60E+05	4,82E+03
100	5,00%	1,60E+05	1,76E+04
100	8,00%	1,60E+05	2,72E+04
100	9,00%	1,60E+05	3,04E+04
100	46,00%	1,60E+05	1,49E+05
100	56,00%	1,60E+05	1,81E+05
200	0,00%	6,40E+05	3,22E+03
200	1,00%	6,40E+05	1,60E+04
200	5,00%	6,40E+05	6,72E+04
200	8,00%	6,40E+05	1,06E+05
200	9,00%	6,40E+05	1,18E+05
200	46,00%	6,40E+05	5,92E+05
200	56,00%	6,40E+05	7,20E+05
300	0,00%	1,44E+06	4,82E+03
300	1,00%	1,44E+06	3,36E+04
300	5,00%	1,44E+06	1,49E+05
300	8,00%	1,44E+06	2,35E+05
300	9,00%	1,44E+06	2,64E+05
300	46,00%	1,44E+06	1,33E+06
300	56,00%	1,44E+06	1,62E+06
400	0,00%	2,56E+06	6,42E+03
400	1,00%	2,56E+06	5,76E+04
400	5,00%	2,56E+06	2,62E+05
400	8,00%	2,56E+06	4,16E+05
400	9,00%	2,56E+06	4,67E+05
400	46,00%	2,56E+06	2,36E+06
400	56,00%	2,56E+06	2,87E+06
500	0,00%	4,00E+06	8,02E+03
500	1,00%	4,00E+06	8,80E+04
500	5,00%	4,00E+06	4,08E+05
500	8,00%	4,00E+06	6,48E+05
500	9,00%	4,00E+06	7,28E+05
500	46,00%	4,00E+06	3,68E+06
500	56,00%	4,00E+06	4,48E+06

Сравнение полученных данных.

Размер	Заполнение	Выигрыш разреженных матриц по памяти, %	Выигрыш разреженных матриц по времени, %
100	0,00%	9801	569261
100	1,00%	3222	1321
100	5,00%	808	130
100	8,00%	488	9
100	9,00%	426	-12
100	46,00%	8	-95
100	56,00%	-12	-96
200	0,00%	19800	4214084
200	1,00%	3896	1713
200	5,00%	852	64
200	8,00%	506	-27
200	9,00%	440	-41
200	46,00%	8	-97
200	56,00%	-11	-98
300	0,00%	29800	9809845
300	1,00%	4184	1580
300	5,00%	868	27
300	8,00%	512	-46
300	9,00%	445	-56
300	46,00%	8	-98
300	56,00%	-11	-98
400	0,00%	39800	14576648
400	1,00%	4343	1420
400	5,00%	876	4
400	8,00%	515	-56
400	9,00%	448	-65
400	46,00%	8	-98
400	56,00%	-11	-99
500	0,00%	49800	32520703
500	1,00%	4445	1345
500	5,00%	880	-8
500	8,00%	517	-62
500	9,00%	449	-70
500	46,00%	9	-99
500	56,00%	-11	-99

Разреженные матрицы выигрывают по памяти до заполнения в 50%.

Алгоритм умножения csc-матрицы на csr-матрицу выигрывает по времени, в среднем, при заполнении до 8-10%.

6. Вывод.

Использование разреженных матриц полезно при маленьком проценте заполнения (до 50%). При таких процентах заполнения разреженная матрица занимает меньше памяти, чем при обычном методе хранения.

Умножение csc-матрицы на csr-матрицу эффективнее традиционного умножения при заполнении до 10%. Такой низкий результат может быть обусловлен тем, что порядок, в котором хранятся входные матрицы, не соответствует порядку, в котором происходит традиционное умножение матриц (матрица 1 хранится по столбцам, матрица 2 хранится по строкам), а так же внутри реализованного алгоритма происходит выделение динамической памяти.

7. Контрольные вопросы.

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

   Разреженная матрица – матрица, в которой количество нулевых элементов превосходит количество ненулевых.

   Существует несколько различных схем хранения таких матриц:
   - Линейный связный список, содержащий элемент матрицы и указатель на следующий.
   - Кольцевой связный список.
   - Диагональная схема хранения симметричных матриц.
   - Разреженный строчный, столбцовый форматы (csr, csc).

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

   Под обычную матрицу выделяется объём памяти, пропорциональный произведению количества строк и количества матрицы.
   Под разреженную матрицу в csc или csr-формате выделяется объём памяти равный сумме размеров трех векторов, из которых она состоит.
   Размер 1 и 2 вектора пропорционален количеству ненулевых матриц.
   Размер 3 вектора пропорционален количеству строк или столбцов матрицы (строк при хранении в csr формате, столбцов – в csc формате).

3. Каков принцип обработки разреженной матрицы?

   При обработке разреженной матрицы используется вектор указателей на начало описания очередной строки/столбца матрицы, при помощи элементов которого происходит обход ненулевых элементов матрицы по оставшимся векторам.

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

   Стандартные алгоритмы более эффективны при большем проценте заполнения матрицы.