

Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное
учреждение высшего образования «Национальный
исследовательский ядерный университет «МИФИ»

Институт ИИКС
Кафедра компьютерных систем и технологий

Лабораторная работа №1:

«Последовательная реализация»

по дисциплине

«Гибридные суперкомпьютерные технологии»

Выполнил:
студент гр. М21-502
Корнилов А. Н.

Проверил:
Синельников Дмитрий Михайлович

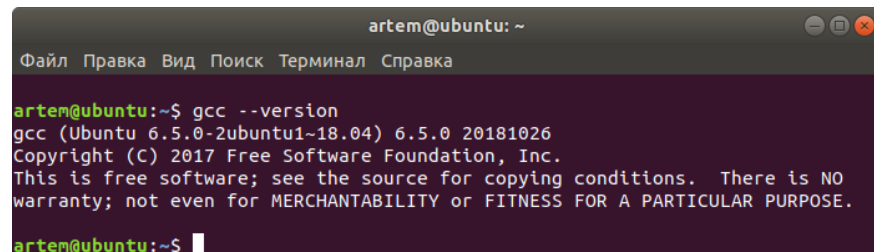
Москва, 2022 г.

Оглавление

Средства разработки	3
Инструкция выполнения	3
Описание компонентов	4
Описание алгоритма:	5
Тесты	7
Результаты	9
График зависимости	11
Приложение А – текст программы «Генератор N матриц»	12
Приложение Б – текст программы «Шифратор матриц»	13

Средства разработки

Разработка программ велась на языке Си с использованием компилятора gcc (см. рисунок 1) и текстового редактора Visual Studio Code на OS Ubuntu.

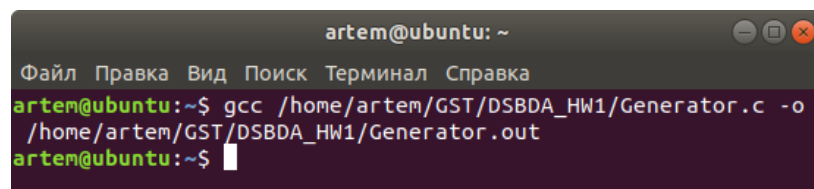


```
artem@ubuntu: ~  
Файл Правка Вид Поиск Терминал Справка  
artem@ubuntu:~$ gcc --version  
gcc (Ubuntu 6.5.0-2ubuntu1~18.04) 6.5.0 20181026  
Copyright (C) 2017 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
artem@ubuntu:~$
```

Рисунок 1 – скриншот версии gcc

Инструкция выполнения

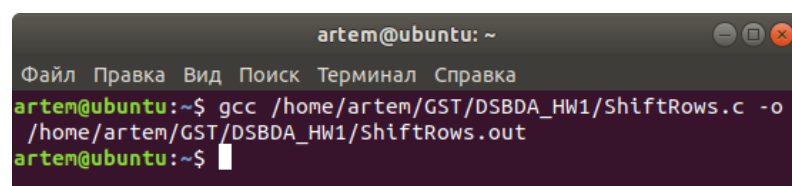
1. Компиляция генератора (см. рисунок 2);



```
artem@ubuntu: ~  
Файл Правка Вид Поиск Терминал Справка  
artem@ubuntu:~$ gcc /home/artem/GST/DSBDA_HW1/Generator.c -o  
/home/artem/GST/DSBDA_HW1/Generator.out  
artem@ubuntu:~$
```

Рисунок 2 – компиляция первой части программы

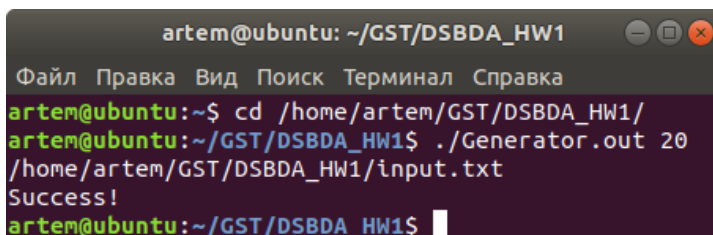
2. Компиляция шифратора (см. рисунок 3);



```
artem@ubuntu: ~  
Файл Правка Вид Поиск Терминал Справка  
artem@ubuntu:~$ gcc /home/artem/GST/DSBDA_HW1/ShiftRows.c -o  
/home/artem/GST/DSBDA_HW1/ShiftRows.out  
artem@ubuntu:~$
```

Рисунок 3 – компиляция второй части программы

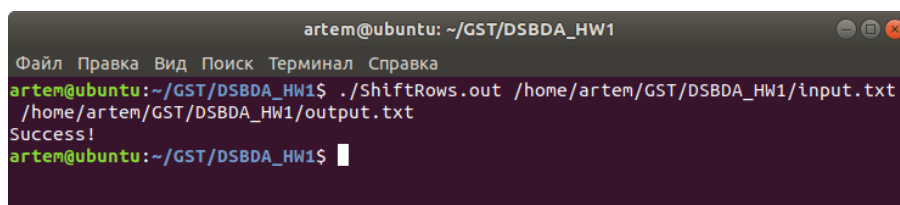
3. Запуск генератора (см. рисунок 4);



```
artem@ubuntu: ~/GST/DSBDA_HW1
Файл Правка Вид Поиск Терминал Справка
artem@ubuntu:~$ cd /home/artem/GST/DSBDA_HW1/
artem@ubuntu:~/GST/DSBDA_HW1$ ./Generator.out 20
/home/artem/GST/DSBDA_HW1/input.txt
Success!
artem@ubuntu:~/GST/DSBDA_HW1$
```

Рисунок 4 – выполнение первой части программы

4. Запуск шифратора (см. рисунок 5)



```
artem@ubuntu: ~/GST/DSBDA_HW1
Файл Правка Вид Поиск Терминал Справка
artem@ubuntu:~/GST/DSBDA_HW1$ ./ShiftRows.out /home/artem/GST/DSBDA_HW1/input.txt
/home/artem/GST/DSBDA_HW1/output.txt
Success!
artem@ubuntu:~/GST/DSBDA_HW1$
```

Рисунок 5 – выполнение первой части программы

Описание компонентов

Программа состоит из двух частей:

1. Файл-генератор N-матриц (где N – первый введенный пользователем аргумент при запуске исполняемого файла), генерирует N прямоугольных матриц со случайной размерностью (каждая матрица имеет свою размерность, т. е. количество строк и столбцов) и случайным элементами (числа от 1 до 999) и записывает результаты в выходной файл (путь к которому должен быть прописан пользователем как второй аргумент). Если не указано число матриц указано как натуральное (начиная от единицы и выше) и файл по

указанному пути существует и имеет право на запись, то помимо записанных матриц и их размерностей в файл в консоль будет выведено «Success!». В противном случае программа оповестит пользователя служебными сообщениями о соответствующих ошибках.

2. Файл-шифратор: шифрует записанные во входном файле (первый введенный пользователем аргумент при запуске исполняемого файла) N матриц, считывает данные о матрицах и их размерностях, выполняет операцию ShiftRows над строками матрицы и записывает получившиеся результаты в выходной файл (второй введенный пользователем аргумент). В случае ненахождения программой входного и выходного файлов или невозможности их открытия (по причине отсутствия прав на чтение входного файла или запись данных в выходной файл) пользователь будет оповещен соответствующими служебными сообщениями об ошибках.

Описание алгоритма:

Алгоритм выполнения бизнес-логики:

1. Фиксирование времени начала выполнения основной программы в миллисекундах;
2. Считывание информации о размерности матрицы из входного файла в строку;
3. Преобразование считанной строки в массив целых чисел;
4. Обход матрицы (выполняется до последней матрицы включительно, т. е. до встречи конца файла):
 - 4.1. Обход матрицы по строкам (от начальной до конечной):

- 4.1.1. Обход матрицы по столбцам (от столбца с номером «количество столбцов в матрице минус остаток от деления текущей строки на количество столбцов» до столбца с номером «количество столбцов в матрице»): запись элемента текущего столбца старой матрицы в новую матрицу на «реальную» позицию столбца (от первого столбца до столбца с номером «количество столбцов в матрице минус остаток от деления текущей строки на количество столбцов»);
- 4.1.2. Обход матрицы по столбцам (от первого столбца до столбца с номером «количество столбцов в матрице минус остаток от деления текущей строки на количество столбцов»): запись элемента текущего столбца старой матрицы в новую матрицу на «реальную» позицию столбца (от столбца с номером «количество столбцов в матрице минус остаток от деления текущей строки на количество столбцов» до последнего столбца в матрице);
- 4.2. Запись элементов новой матрицы и её размерности в файл построчно;
- 4.3. Вычисление размера обработанных данных в мегабайтах и его запись в выходной файл;
- 4.4. Фиксирование времени окончания выполнения основной программы в миллисекундах;
- 4.5. Вычисление разности времени окончания и начала выполнения основной программы в миллисекундах и её запись в файл.

На рисунке 6 представлена иллюстрация выполнения операции ShiftRows над прямоугольной матрицей размерности 4x4.

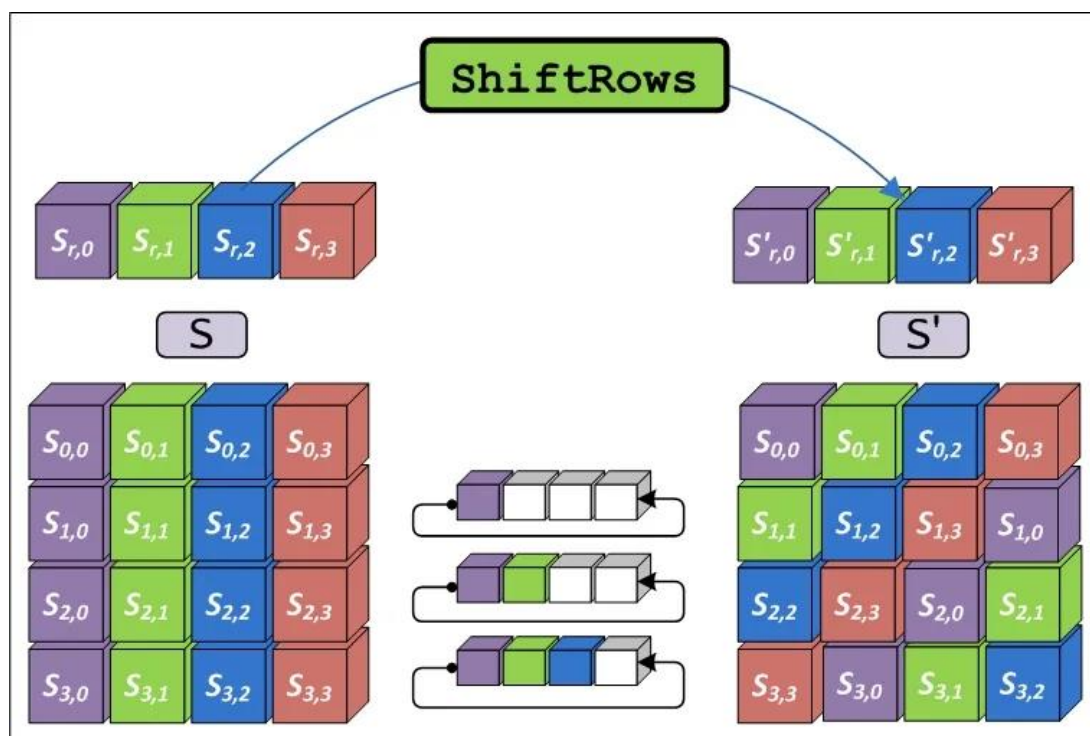


Рисунок 6 – выполнение операции ShiftRows над матрицей

Тесты

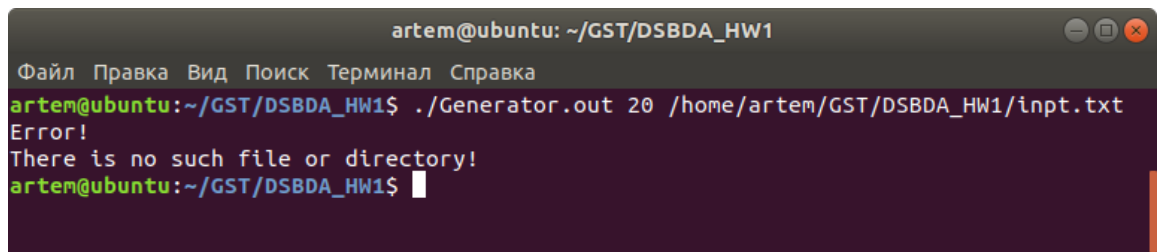
Результат реагирования программы на различные ошибки представлены на рисунках с 7 по 10.

```

artem@ubuntu: ~/GST/DSBDA_HW1
Файл Правка Вид Поиск Терминал Справка
artem@ubuntu:~/GST/DSBDA_HW1$ ./Generator.out /home/artem/GST/DSBDA_HW1/input.txt
Error!
You didn't write number of matrices or outut file path!
artem@ubuntu:~/GST/DSBDA_HW1$

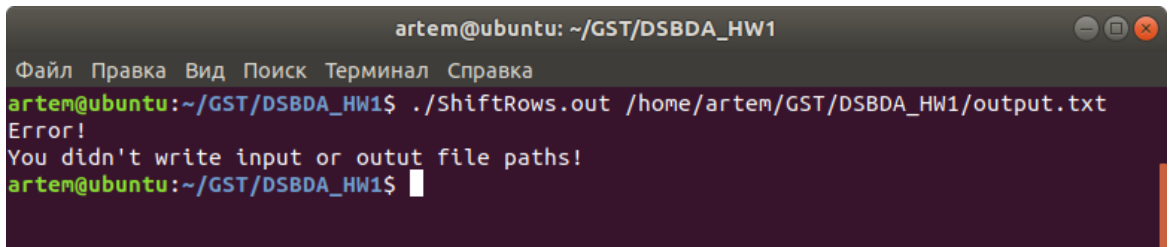
```

Рисунок 7 – ошибка «Не задано количество матриц»



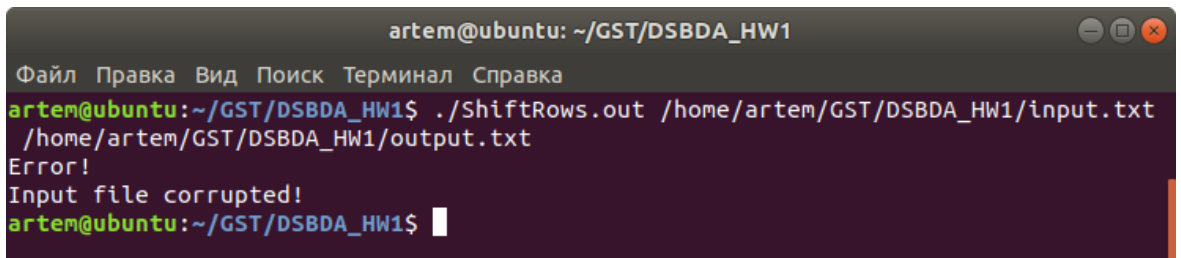
```
artem@ubuntu: ~/GST/DSBDA_HW1
Файл Правка Вид Поиск Терминал Справка
artem@ubuntu:~/GST/DSBDA_HW1$ ./Generator.out 20 /home/artem/GST/DSBDA_HW1/inpt.txt
Error!
There is no such file or directory!
artem@ubuntu:~/GST/DSBDA_HW1$
```

Рисунок 8 – ошибка «Не удалось найти файл по указанному каталогу»



```
artem@ubuntu: ~/GST/DSBDA_HW1
Файл Правка Вид Поиск Терминал Справка
artem@ubuntu:~/GST/DSBDA_HW1$ ./ShiftRows.out /home/artem/GST/DSBDA_HW1/output.txt
Error!
You didn't write input or output file paths!
artem@ubuntu:~/GST/DSBDA_HW1$
```

Рисунок 9 – ошибка «Не введен путь к входному или выходному файлу»

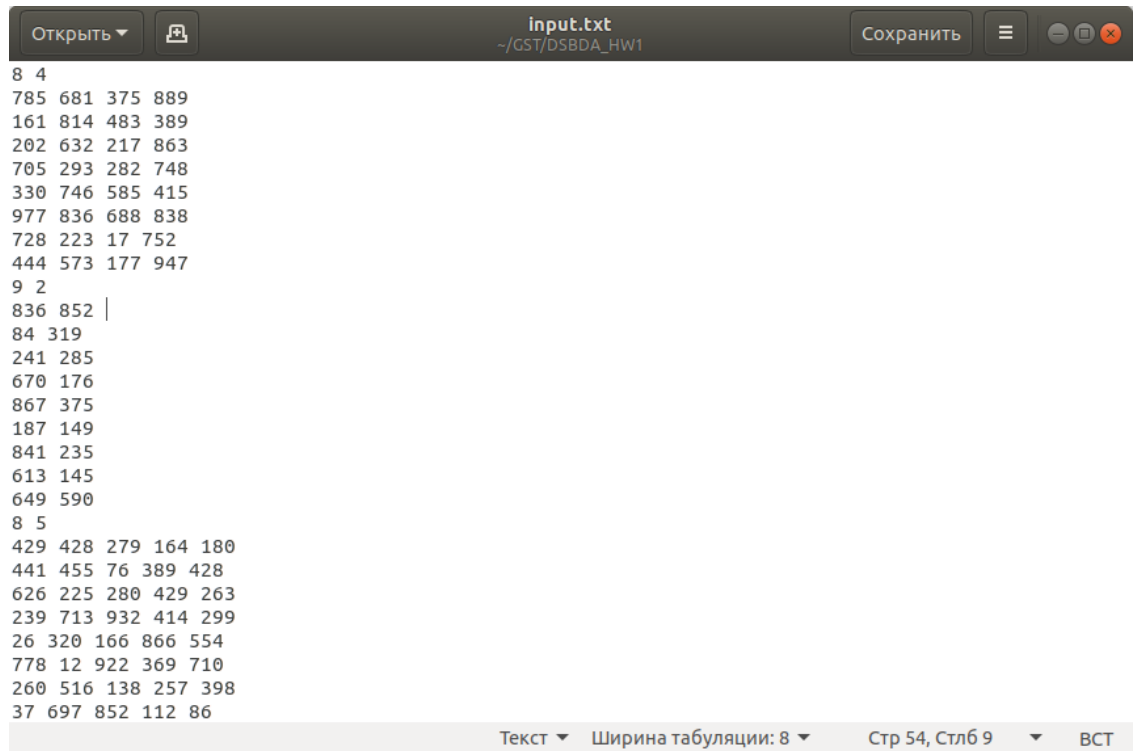


```
artem@ubuntu: ~/GST/DSBDA_HW1
Файл Правка Вид Поиск Терминал Справка
artem@ubuntu:~/GST/DSBDA_HW1$ ./ShiftRows.out /home/artem/GST/DSBDA_HW1/input.txt
/home/artem/GST/DSBDA_HW1/output.txt
Error!
Input file corrupted!
artem@ubuntu:~/GST/DSBDA_HW1$
```

Рисунок 10 – ошибка «Входной файл повреждён»

Результаты

На рисунке 11 представлен пример сгенерированных данных.



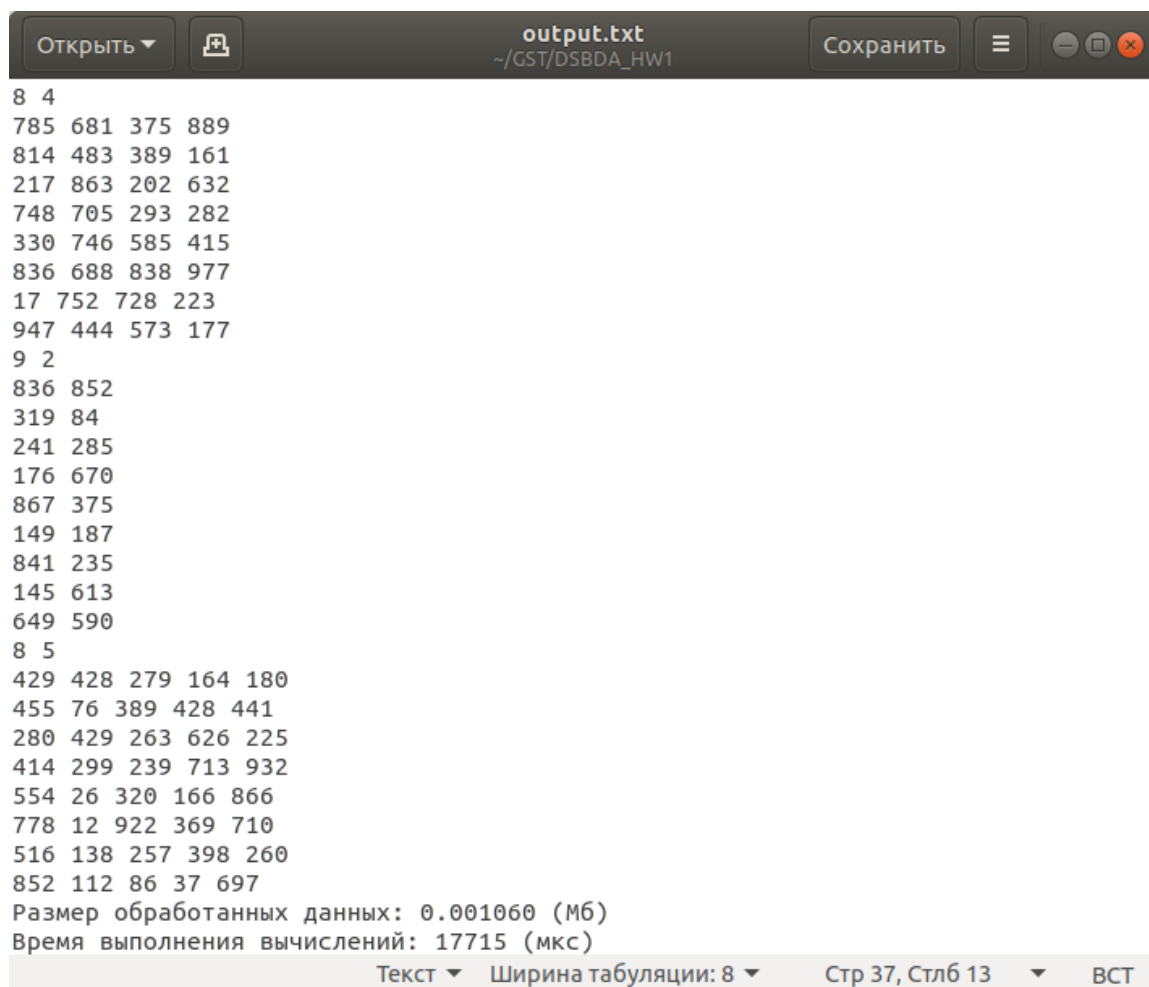
The screenshot shows a text editor window titled "input.txt" with the path "~/GST/DSBDA_HW1". The window contains two blocks of generated data matrices. The first block is a 9x4 matrix, and the second block is a 9x5 matrix. The data is displayed as a list of rows, each containing a row index followed by the matrix elements.

```
8 4
785 681 375 889
161 814 483 389
202 632 217 863
705 293 282 748
330 746 585 415
977 836 688 838
728 223 17 752
444 573 177 947
9 2
836 852 |
84 319
241 285
670 176
867 375
187 149
841 235
613 145
649 590
8 5
429 428 279 164 180
441 455 76 389 428
626 225 280 429 263
239 713 932 414 299
26 320 166 866 554
778 12 922 369 710
260 516 138 257 398
37 697 852 112 86
```

The status bar at the bottom indicates the file is in "Текст" (Text) mode, with a tab width of 8, and the cursor is at "Стр 54, Стлб 9" (Line 54, Column 9).

Рисунок 11 – сгенерированные матрицы

На рисунке 12 представлен пример зашифрованных данных.



The screenshot shows a text editor window titled "output.txt" with the path "~/GST/DSBDA_HW1". The window contains a table of encrypted data. The table has two main sections, one with 4 columns and one with 5 columns. The data is presented as rows of numbers. At the bottom of the window, there is a status bar showing "Текст", "Ширина табуляции: 8", "Стр 37, Стлб 13", and "ВСТ".

```
8 4
785 681 375 889
814 483 389 161
217 863 202 632
748 705 293 282
330 746 585 415
836 688 838 977
17 752 728 223
947 444 573 177
9 2
836 852
319 84
241 285
176 670
867 375
149 187
841 235
145 613
649 590
8 5
429 428 279 164 180
455 76 389 428 441
280 429 263 626 225
414 299 239 713 932
554 26 320 166 866
778 12 922 369 710
516 138 257 398 260
852 112 86 37 697
Размер обработанных данных: 0.001060 (Мб)
Время выполнения вычислений: 17715 (мкс)
```

Текст ▾ Ширина табуляции: 8 ▾ Стр 37, Стлб 13 ▾ ВСТ

Рисунок 12 – результаты шифрования

График зависимости

В таблице 1 представлены замеры времени выполнения и соответствующие им размеры данных.

Таблица 1 – замеры параметров

Размеры данных, Мб	Время выполнения, с
10.133363	3.971051
20.289106	6.903115
40.552288	12.803378
81.058136	26.483021
162.139465	51.959678
324.382111	103.389792

На рисунке 13 представлен график зависимости времени выполнения от размера данных.

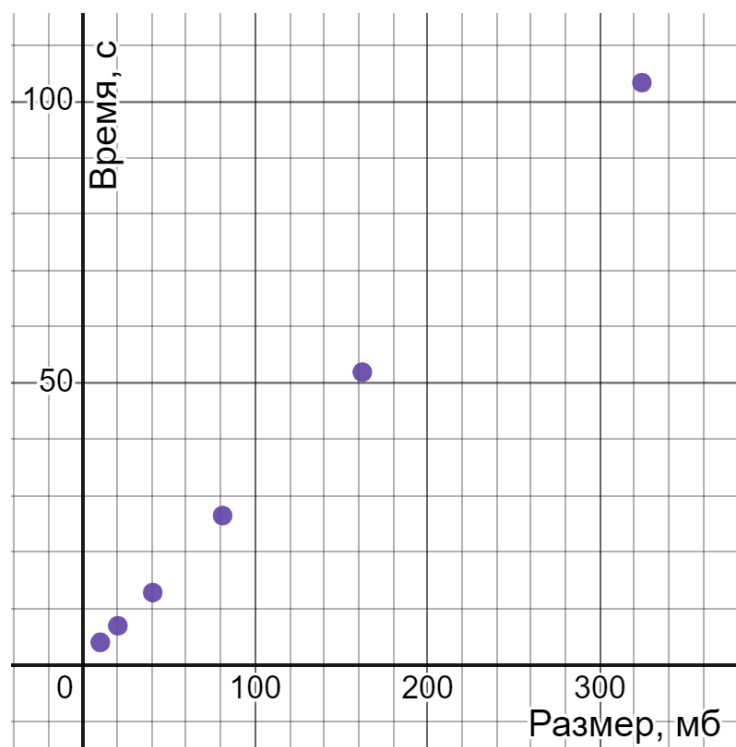


Рисунок 13 – график зависимости времени от размера

Приложение А – текст программы «Генератор N матриц»

```
#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <time.h>

int main(int argc, char *argv[]){

    if (argc < 3){

        printf("Error!\nYou didn't write number of matrices or outut file path!\n");

        exit(0);

    }

    int count_of_matrices = 0;

    if ((count_of_matrices = atoi(argv[1])) < 1){

        printf("Error!\nCount of matrices must be more than 0!\n");

        exit(0);

    }

    int fd = 0;

    if ((fd = open(argv[2], O_WRONLY | O_TRUNC)) < 0){

        printf("Error!\nThere is no such file or directory!\n");

        exit(0);

    }

    srand(time(NULL));

    int out = dup(1);

    dup2(fd, 1);

    for (int number_of_matrix = 0; number_of_matrix < count_of_matrices; number_of_matrix++){

        int count_of_rows = rand()%9 + 1;

        int count_of_columns = rand()%9 + 1;

        printf("%d %d\n", count_of_rows, count_of_columns);

        for (int row = 0; row < count_of_rows; row++){

            for (int column = 0; column < count_of_columns; column++){
```

```

        printf("%d ", rand()%999 + 1);
    }
    printf("\n");
}
}
write(out, "Success!\n", 9);
}

```

Приложение Б – текст программы «Шифратор матриц»

```

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>

int main(int argc, char *argv[]){
    int start_time = clock();
    if (argc < 3){
        printf("Error!\nYou didn't write input or output file paths!\n");
        exit(0);
    }
    int fd_input = 0;
    if ((fd_input = open(argv[1], O_RDONLY)) < 0){
        printf("Error!\nThere is no such file or directory!\n");
        exit(0);
    }
    int fd_output = 0;
    if ((fd_output = open(argv[2], O_WRONLY | O_TRUNC)) < 0){
        printf("Error!\nThere is no such file or directory!\n");
        exit(0);
    }
}

```

```

int out = dup(1);
dup2(fd_output, 1);
int end_position = lseek(fd_input, 0, SEEK_END);
lseek(fd_input, 0, SEEK_SET);
while (lseek(fd_input, 0, SEEK_CUR) != end_position){
    char dimension[5];
    read(fd_input, dimension, 4);
    char rows_buffer[2];
    rows_buffer[0] = dimension[0];
    char columns_buffer[2];
    columns_buffer[0] = dimension[2];
    int count_of_rows = atoi(rows_buffer);
    int count_of_columns = atoi(columns_buffer);
    if (count_of_rows == 0 || count_of_columns == 0){
        write(out, "Error!\nInput file corrupted!\n", 29);
        exit(0);
    }
    printf("%d %d\n", count_of_rows, count_of_columns);
    int new_matrix[count_of_rows][count_of_columns];
    for (int row = 0; row < count_of_rows; row++){
        for (int column = count_of_columns - (row%count_of_columns); column < count_of_columns;
column++){
            char current_element[5];
            read(fd_input, current_element, 4);
            new_matrix[row][column] = atoi(current_element);
            if (new_matrix[row][column] == 0){
                write(out, "Error!\nInput file corrupted!\n", 29);
                exit(0);
            }
        }
        int bit_depth = 100;
        while(1){
            if (new_matrix[row][column]/bit_depth == 0){

```

```

        bit_depth /= 10;

        lseek(fd_input, -1, SEEK_CUR);
    }
    else break;
}

}

for (int column = 0; column < count_of_columns - (row%count_of_columns); column++){
    char current_element[5];
    read(fd_input, current_element, 4);
    new_matrix[row][column] = atoi(current_element);
    if (new_matrix[row][column] == 0){
        write(out, "Error!\nInput file corrupted!\n", 29);
        exit(0);
    }
    int bit_depth = 100;
    while(1){
        if (new_matrix[row][column]/bit_depth == 0){
            bit_depth /= 10;
            lseek(fd_input, -1, SEEK_CUR);
        }
        else break;
    }
}

for (int column = 0; column < count_of_columns; column++){
    printf("%d ", new_matrix[row][column]);
}

printf("\n");

lseek(fd_input, 1, SEEK_CUR);
}

}

write(out, "Success!\n", 9);

printf("Размер обработанных данных: %f (M6)\n", (((float) end_position) / 1024.0) / 1024.0));

```

```
int finish_time = clock();  
printf("Время выполнения вычислений: %d (мс)\n", finish_time - start_time);  
}
```