ФГОБУ ВО «Финансовый университет при Правительстве Российской Федерации»
ВСЕРОССИЙСКАЯ ОЛИМПИАДА ПО ИНФОРМАТИКЕ
«МИССИЯ ВЫПОЛНИМА. ТВОЕ ПРИЗВАНИЕ – ФИНАНСИСТ!»
ОЧНЫЙ ЭТАП, 2026 год

Продолжительность олимпиады – 235 минут. Олимпиадное задание состоит из пяти задач. Для каждой задачи указан ее вес в баллах.
Участник олимпиады самостоятельно определяет последовательность выполнения задач. На одном из языков программирования – C/C++, C#, Visual Basic, Pascal или Python – разработайте *консольные* программы для решения перечисленных ниже задач.

При выполнении задания участник формирует каталог в имени которого указывает свое ФИО. В данном каталоге формирует пять каталогов: Task1; Task2; Task3; Task4; Task5. Решение задачи размещаются в каталоге с соответствующим номером (см. рис П.1**)**

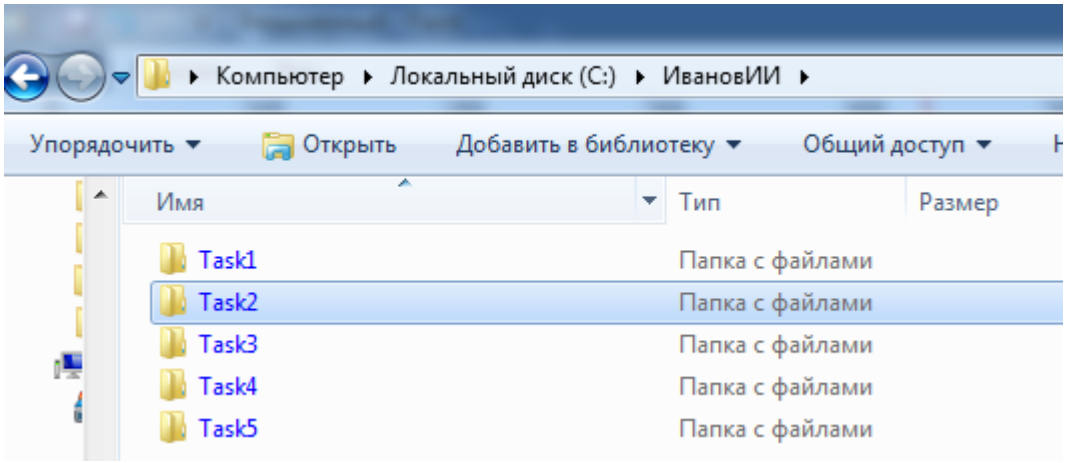


Рисунок П.1 – Структура каталога участника олимпиады

Участник олимпиады должен предоставить членам комиссии на проверку только файлы с исходными текстами программ, которые должны быть названы участником олимпиады в соответствии с выполняемым заданием, например, для языка Python: Task1.py.

Расширение файла должно соответствовать языку. Переименуйте файлы перед сдачей работы, если это необходимо. (см пример на рис. П.2)

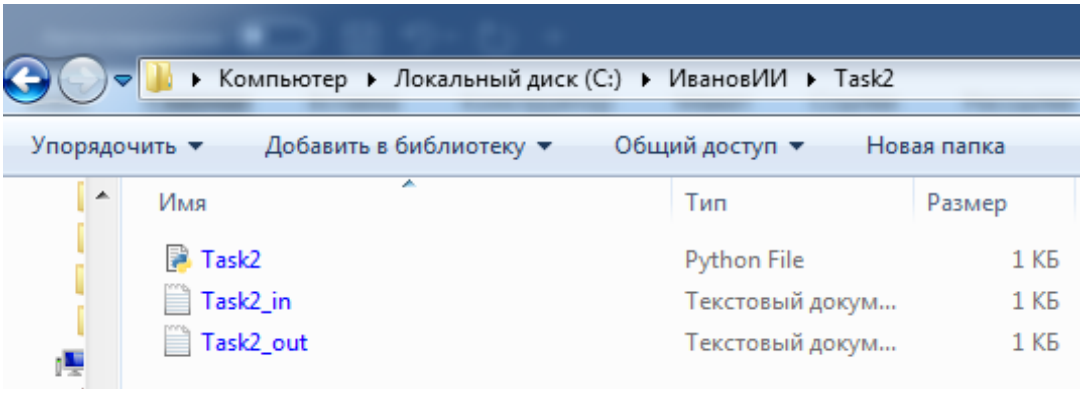


Рисунок П.2 – Размещение файлов в рамках папки задачи

В начале каждой программы должен находиться комментарий с ФИО участника, вариант, номером задачи, языком программирования, средой программирования или

онлайн-компилятора. Например, для С-подобных языков: // Иванов И. И., вариант 1, задача 1, Python 3.7.3, Spyder 3.3.6.

Если файлы с решением задачи, исходных и результирующих данных имеют некорректные названия и/или отсутствует первая строка комментарий, и/или размещены в каталоге участника без учета требований к структуре, то члены комиссии данное решение не оценивают и баллы за решение задания не начисляются.

При решении задач в качестве файлов с исходными данными и выходными данными используется только текстовый файл с расширением *.txt. Если в задаче программной реализации используется файлы с исходными данными и/или выходными данными, то кроме файла с исходным текстом требуется выслать соответствующие файлы. Например, для задания 2 требуется использовать исходные данные из файла, тогда название файла должно быть Task2_in.txt. если в задание 2 требуется сформировать текстовый файл с результатами исполнения программы, то название файла должно быть Task2_out.txt. Число после слова Task соответствует номеру решенного задания, «_in» определяет, что файл с исходными данными, «_out» определяет, что в файле хранятся результаты исполнения кода над исходными данными. Все текстовые файлы с исходными данными создаются участником самостоятельно, в соответствии с представленными в задачах примерами и шаблонами. Ввод и вывод текста осуществляется только латинскими буквами.

По окончанию работы над заданиями участник формирует архив с содержимым решений в соответствии с предложенной выше структурой. Расширение архивного файла должно быть rar или zip (пример см. рис П.3-П.5).

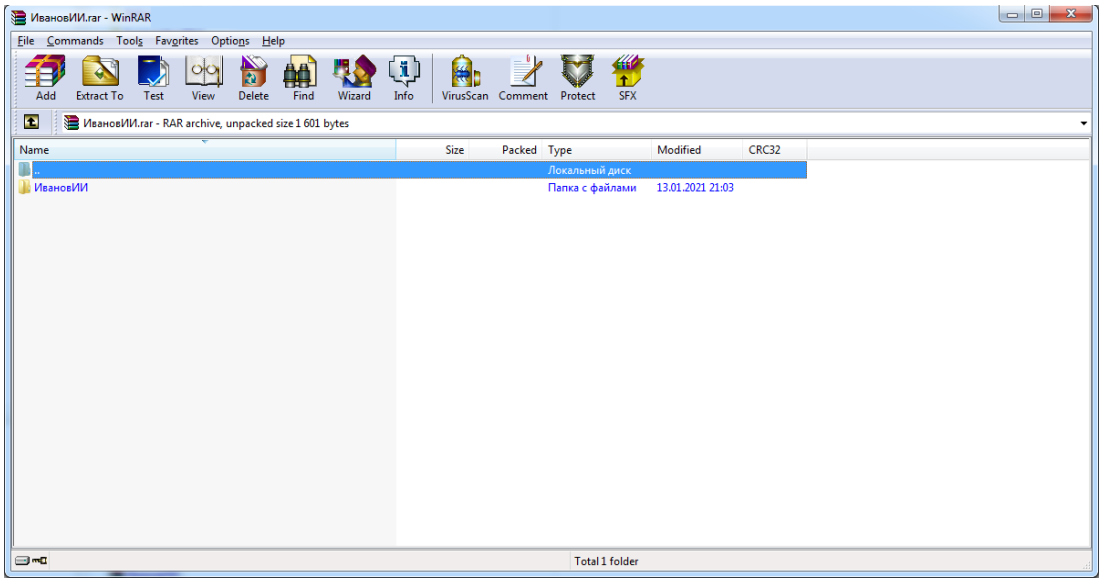


Рисунок П.3 – Пример первого уровня содержимого архива

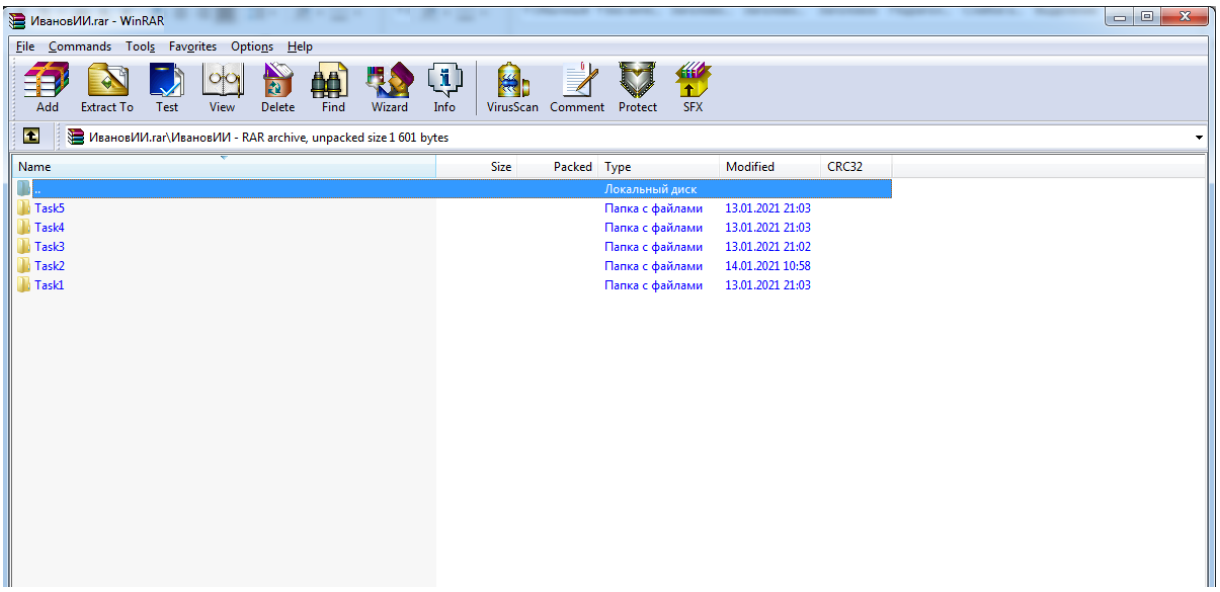


Рисунок П.3 – Пример второго уровня содержимого архива

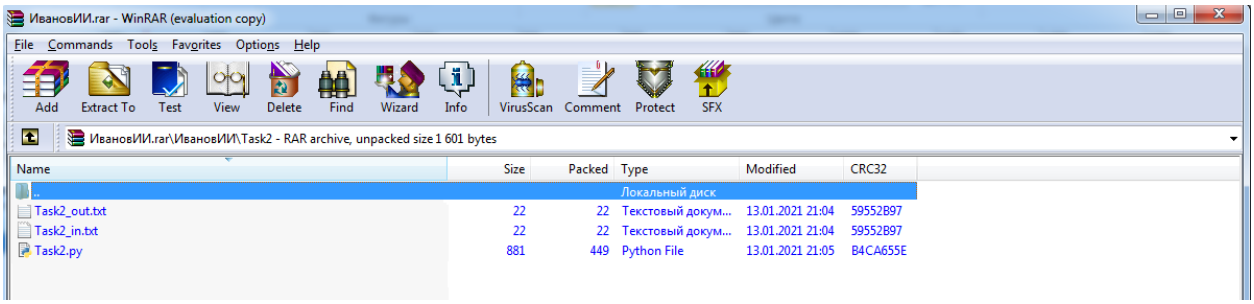


Рисунок П.3 – Пример третьего уровня содержимого архива

Члены комиссии, при проверке заданий, используют операционные системы семейства Windows (версии 7-11). Если члену комиссии не удается запустить файл с исходным кодом на исполнение (например, программа не выполняется в перечисленных ОС и/или не находит файл с исходными данными, и/или не формирует результирующий файл и т. д.), то задание считается не выполненным.

Максимальное количество баллов, которые может набрать участник – 100.

При оценивании решения задачи члены жюри могут снизить баллы за следующие недостатки:

- неполное соответствие решения условию;
- применение неэффективного алгоритма;
- решение задачи только для частного случая;
- отсутствие проверок, приводящих к снижению надежности программы;
- низкое качество интерфейса пользователя;
- несоответствие решения пулу тестовых значений;
- плохая читабельность текста программы и т. д.

При обнаружении использования участником: посторонней помощи в любом проявлении, средств интернет, мобильных устройств и других приемо-передающих устройств, способствующих решению заданий олимпиады, не используя собственные знания, приведут к исключению участника и аннулированию его результатов.

Для программной реализации заданий участник олимпиады должен использовать онлайн-компилятор https://www.onlinegdb.com/, пройдя процедуру регистрации.

При неработоспособности онлайн-компилятора https://www.onlinegdb.com/ участник олимпиады может использовать следующие среды разработки: CodeBlocks, Anaconda, Lazarus, Mono + MonoDevelop, установленные на ЭВМ.

В случае, если программный код участника не запускается в вышеперечисленных средах разработки, комиссия не рассматривает данную работу.

Задания на очный этап олимпиады «Информатика»
Вариант №2

Задание 1 (8 баллов)

Реализовать программу для расчета функции y=y(x), представленной на рисунке 1.1. Значения x вводятся с клавиатуры, результат вычислений выводится на экран ПК.

$$y = \frac{x-1}{x^2+2x+1} + \ln(x+5)$$

Рисунок 1.1 — Функция y=y(x)

Задание 2 (12 баллов)

Выполнить программную реализацию расчета значений логической функции четырех переменных (см. рис. 2.1) и вывод результата в файл Task2_out.txt.

$$F = (A\ and\ B)\ and\ (C\ or\ D) \rightarrow notA$$

Рисунок 2.1 – Логическая функция четырех переменных

В функции обозначения соответствуют:

- *or* – логическое ИЛИ (дизъюнкция);
- and – логическое И (конъюнкция);
- *not* – логическое отрицание (НЕ);
- → – импликация.

Исходные данные с 16 комбинациями значений переменных требуется считать из файла Task2_in.txt. (пример см. рис. 2.2, порядок столбцов A B C D всегда соответствует данному примеру и не может быть изменен).

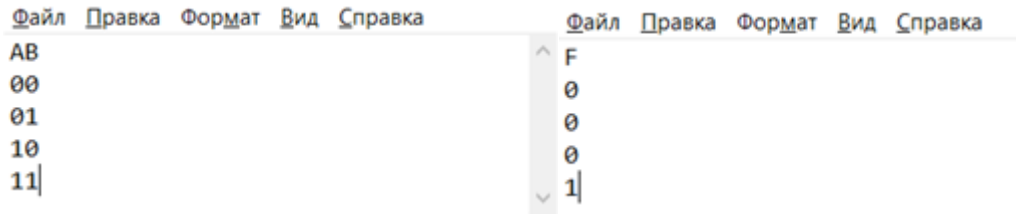
Файл Правка Формат Вид Справка
ABCD
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Рисунок 2.2 – Файл Task2_in.txt

Значение F вычисляется разработанной программой (за ручное решение с записью в файл Task2_out.txt баллы не начисляются), и записывается в файл Task2_out.txt.

На рисунках 2.3 a) и 2.3 b) показаны примеры файлов Task2_in.txt и Task2_out.txt, заполненных в соответствии с требованиями, для логической функции от двух переменных

F = A and B.



a) b)

Рисунок 2.3 – a) Файл Task2_in.txt, b) Файл Task2_out.txt

Задание 3 (20 баллов)

В переписке между филиалами Финуниверситета для передачи важных сообщений решили использовать шифр. Длина сообщений при этом никогда не превышает 26 символов, включая пробелы.

Процесс шифрования начинается с построения сетки размером 5×5, где каждая ячейка заполняется 25 буквами латинского алфавита (буквы I и J шифруются как I). Каждая строка и столбец сетки задаётся одной из 5 цифр: «1», «2», «3», «4» и «5».

	1	**2**	**3**	**4**	**5**
1	A	B	C	D	E
2	F	G	H	I	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Рисунок 3.1

Рассмотрим процесс шифрования на примере небольшого сообщения: «SOMETEXT».

На первом шаге каждый символ сообщения заменяется на пару координат, обозначающих строку и столбец соответствующего символа в сетке:

Сообщение	S	O	M	E	T	E	X	T
Координата по вертикали	3	4	2	5	4	5	3	4
Координата по горизонтали	4	3	3	1	4	1	5	4

Рисунок 3.2

На втором шаге координаты считываются по строкам:
3425453443314154.

На третьем шаге полученная последовательность сдвигается циклически вправо на один шаг:
4342545344331415.

На четвертом шаге координаты преобразуются в буквы по сетке, представленной на рисунке 3.1:

Координата по вертикали	4	4	5	5	4	3	1	1
Координата по горизонтали	3	2	4	3	4	3	4	5
Сообщение	O	I	U	P	T	N	Q	V

Рисунок 3.3

Окончательный вид шифротекста: OIUPTNQV.

Для восстановления исходного текста необходимо выполнить действия, обратные шифрованию.

Необходимо автоматизировать процесс шифрования и дешифрования с помощью описанного метода (пробелы в сообщениях не шифруются и соответственно не дешифруются).

Требования к программе:

1. Пользователь выбирает действие, которое собирается осуществить (реализация меню см. рис. 3.4):

```
Выберите действие:
Шифрование нажмите - 1
Дешифрование нажмите - 2
```

Рисунок 3.4

2. При нажатии «1» программное средство должно выполнить шифрование.

3. При нажатии «2» программное средство должно выполнить дешифрование.

При шифровании в качестве входного файла программа должна использовать файл Task3_in_SH.txt структура которого представлена на рисунке 3.5.

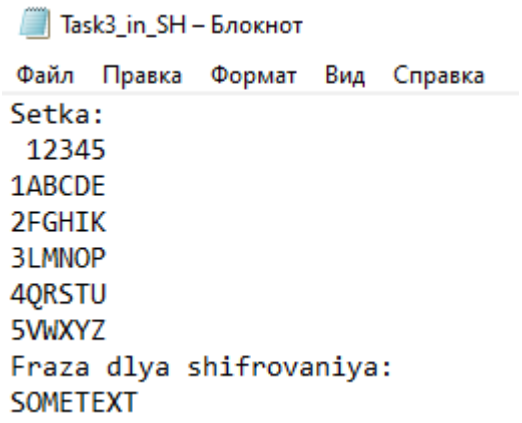


Рисунок 3.5

В качестве выходного файла формируется файл Task3_out_SH.txt структура которого представлена на рисунке 3.6.

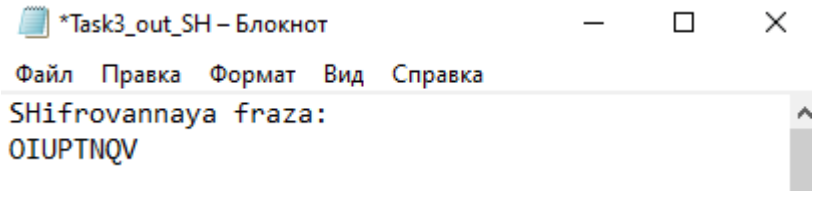


Рисунок 3.6

При дешифровании в качестве входного файла программа должна использовать файл Task3_in_DSH.txt структура которого представлена на рисунке 3.7.

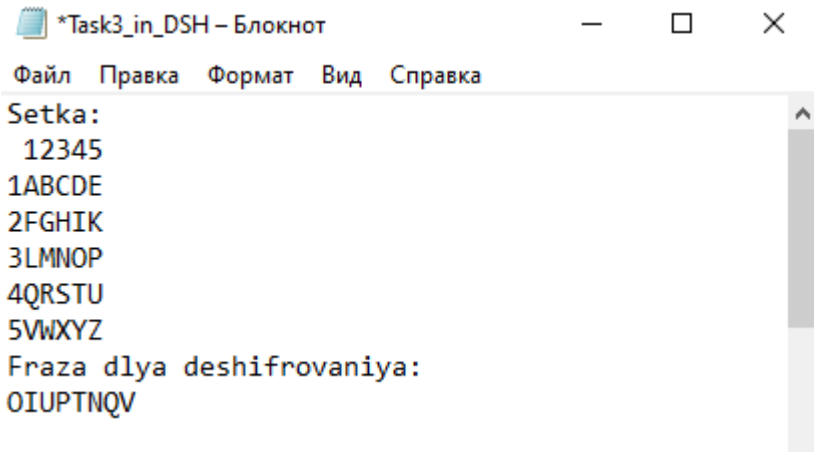


Рисунок 3.7

В качестве выходного файла при дешифровании формируется файл Task3_out_DSH.txt структура которого представлена на рисунке 3.8.

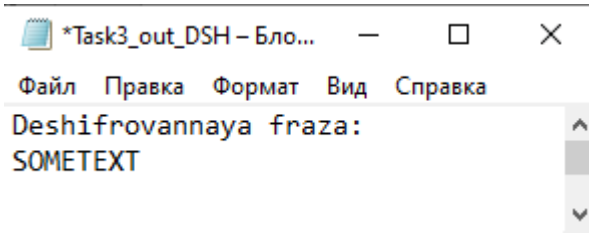


Рисунок 3.8

При шифровании и дешифровании фраз требуется выполнить проверки:

1. На существование в исходном сообщении всех символов, которые будут подвержены шифрованию или дешифрованию (сообщение должно содержать только буквы латинского алфавита и пробелы);
2. Длина сообщения не превышает 26 символов (пробелы из сообщения удаляются или сообщение вводится без пробелов).

При несоответствии требованиям программное средство должно сформировать следующие сообщениях в Task3_out_SH и Task3_out_DSH:

1. «В исходном сообщении указаны символы, которые не могут быть зашифрованы или дешифрованы»;
2. «Длина сообщения не соответствует требованиям».

Задание 4 (25 баллов)

Во время проведения крупных измерительных экспериментов исследовательские группы получают многомерные данные от различных датчиков и устройств. Каждая серия измерений формируется в виде прямоугольной числовой матрицы фиксированного размера. Такие матрицы описывают распределения данных, собранных на единичном наборе наблюдений, и могут отражать температуры, влажности, интенсивности сигналов, уровни шума и другие параметры. Однако в силу особенностей оборудования или внешних факторов некоторые матрицы содержат аномальные измерения – слишком широкие диапазоны значений, выходящие за установленные допустимые пределы. Если матрица демонстрирует чрезмерный разброс, она считается аномальной и исключается из дальнейшего анализа; в противном случае её необходимо нормировать, приводя значения к диапазону от 0 до 1, что позволяет сравнивать разные наборы данных между собой.

Необходимо создать входной файл **Task4_in.txt**, содержащий несколько матриц одинакового размера. Файл формируется вручную и начинается с одной строки, в которой записаны четыре натуральных числа *K*, *N*, *M* и *T*. Здесь *K* обозначает количество матриц, *N* – число строк в каждой матрице, *M* – число столбцов, а *T* – максимально допустимый разброс значений в матрице. После этой строки в файле располагаются *K* последовательных блоков, каждый из которых содержит ровно *N* строк по *M* целых чисел в каждой строке. Таким образом, файл описывает *K* матриц размерности *N*×*M*, записанных одна за другой. При чтении данных программа должна строго проверить, что каждая строка содержит корректное количество чисел, а сами числа могут быть преобразованы к целочисленному типу; любые отклонения от формата считаются ошибкой.

После успешного чтения входного файла программа должна для каждой матрицы определить минимальное и максимальное значение среди её элементов и вычислить их разность. Если эта разность превышает установленный порог *T*, матрица относится к группе *аномальных*. Если же диапазон значений не превышает *T*, матрица считается корректной и подлежит нормировке: каждое значение *x* преобразуется по формуле $(x - \min) / (\max - \min)$. В случае, когда max = min (то есть все элементы матрицы совпадают), нормированная форма представляет собой матрицу из нулей того же размера *N*×*M*. Нормированные значения должны округляться до двух знаков после десятичной точки.

После определения корректных и аномальных матриц программа должна сформировать два выходных файла. В файл **Task4_1_out.txt** записываются все *корректные* матрицы в том же порядке, в котором они были прочитаны из входного файла. Первая строка этого файла должна содержать количество корректных матриц и размерность *N M*. Далее каждая корректная матрица записывается двумя последовательными фрагментами: сначала её исходная форма (*N* строк по *M* целых чисел), а сразу вслед за ней – нормированная матрица того же размера, в которой все значения вычислены по заданной формуле и округлены до двух знаков после запятой. Если корректных матриц нет, файл Task4_1_out.txt должен содержать только строку «*0 N M*».

Файл **Task4_2_out.txt** предназначен для *аномальных* матриц. Его первая строка должна аналогично содержать количество аномальных матриц и размерность *N M*. Затем каждая аномальная матрица приводится в виде *N* строк по *M* целых чисел, в той же последовательности, в которой она была прочитана из входного файла. Никаких нормированных значений или дополнительных комментариев в этот файл не включается. Если аномальных матриц нет, файл Task4_2_out.txt должен содержать единственную строку «*0 N M*».

Ниже приведён пример корректно сформированного входного файла Task4_in.txt, содержащего две матрицы размера 3×3 и значение порога допустимого разброса *T*, равное 5.

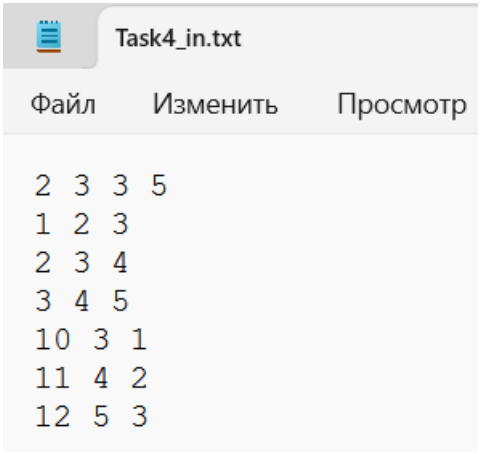


Рисунок 4.1 – Пример входного файла Task4_in.txt

Первая матрица содержит значения от 1 до 5. Минимальный элемент равен 1, максимальный – 5, поэтому разность max – min составляет 4. Поскольку это значение не превышает заданный порог $T = 5$, матрица считается *корректной* и подлежит нормировке. Нормирование выполняется по формуле $(x - \min) / (\max - \min)$; в данном случае получаются значения 0.00, 0.25, 0.50, 0.75 и 1.00, округлённые до двух знаков после десятичной точки. Полученная нормированная матрица имеет тот же размер 3×3, и каждый её элемент представляет собой результат применения указанной формулы к соответствующему элементу исходной матрицы.

Вторая матрица содержит значения от 1 до 12. При минимальном значении 1 и максимальном 12 разность max – min равна 11, что существенно превышает установленный порог $T = 5$. Такая матрица считается *аномальной* и нормировке не подлежит.

Для приведённых входных данных файл Task4_1_out.txt, содержащий корректные матрицы и их нормированные формы, имеет следующий вид (рис. 4.2).

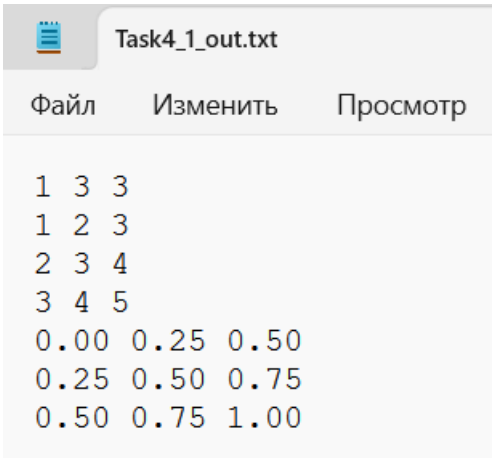


Рисунок 4.2 – Пример файла Task4_1_out.txt

Файл Task4_2_out.txt, в который записываются все аномальные матрицы в их исходном виде, для данного примера формируется следующим образом (рис. 4.3).

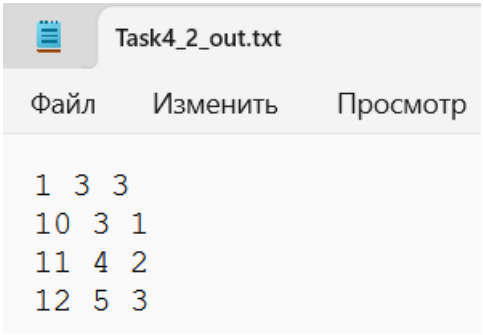


Рисунок 4.3 – Пример файла Task4_2_out.txt

Если во входном файле все матрицы окажутся корректными, то есть ни одна из них не превышает порогового значения *T*, файл Task4_2_out.txt должен состоять из единственной строки, в которой записано количество аномальных матриц (в этом случае 0), а далее указана размерность *N M*. Например, для матриц размером 3×3 содержимое файла будет следующим (рис. 4.4).

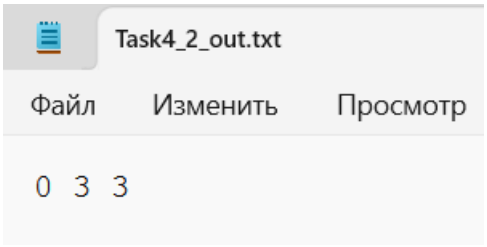


Рисунок 4.4 – Пример файла Task4_2_out.txt при отсутствии аномальных матриц

Аналогично, если во входном файле все матрицы окажутся аномальными, то есть ни одна из них не удовлетворяет требованиям по порогу *T*, файл Task4_1_out.txt должен содержать одну строку, в которой первым числом указано количество корректных матриц (то есть 0), после чего следуют размерности *N* и *M*. Для матриц размерности 3×3 этот файл выглядит так (рис. 4.5).

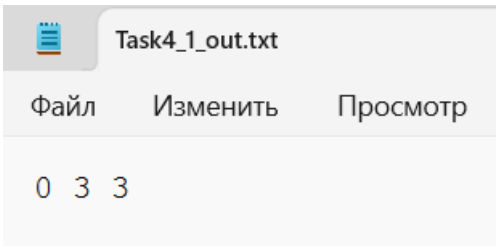


Рисунок 4.5 – Пример файла Task4_1_out.txt при отсутствии корректных матриц

В случае нарушения структуры входного файла, например, если число строк недостаточно, строки содержат неверное количество значений, встречаются нечисловые символы или параметры *K, N, M, T* заданы некорректно, программа должна вывести сообщение «*Ошибка формата данных*» и завершить работу, не создавая выходных файлов.

Задание 5 (35 баллов)

Напишите программу, которая определяет белую фигуру, создавшую положение в шахматной партии «двойной удар с шахом черному королю» после хода белых.

Условия шахматной ситуации:

очередной ход сделан белыми;

на шахматной доске может находится любое допустимое количество черных и белых фигур (пешка, слон, конь, ладья или ферзь), черный король находится обязательно.

Определение:

«Двойной удар» в шахматах или, «вилка» — положение в шахматной партии, когда две или более фигуры одного игрока находятся под боем одной фигуры другого игрока. Если одной из фигур, находящихся под боем, является король, то положение называется «двойной удар с шахом» (пример «двойной удар с шахом» на рисунке 5.1).

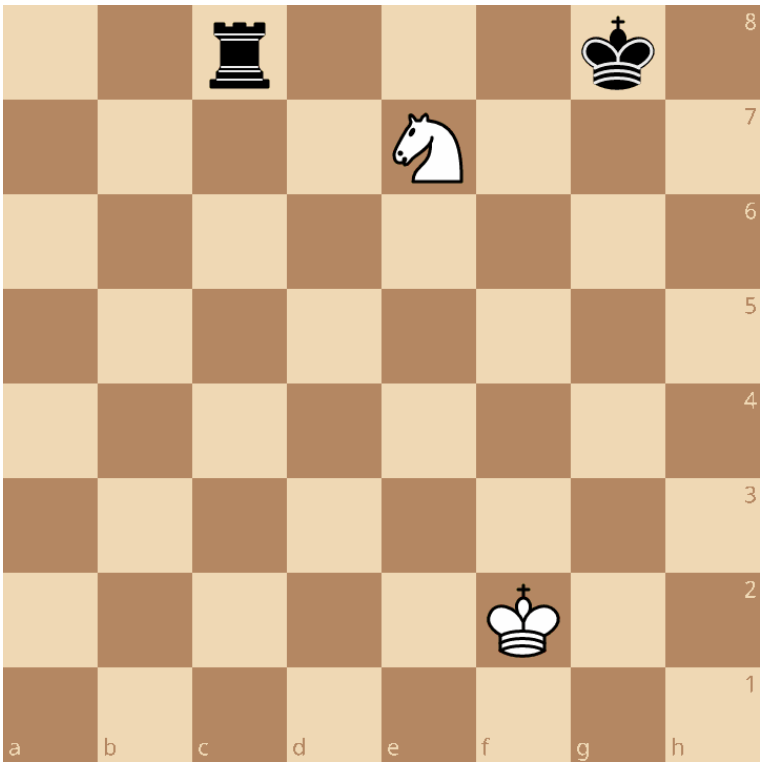


Рисунок 5.1 – Пример положения в шахматной партии «двойной удар с шахом»

Обозначения:

Игровое поле для игры в шахматы представляет собой доску, разделенную на 64 квадратные клетки по 8 клеток по горизонтали и 8 клеток по вертикали. Каждая вертикаль обозначается латинской буквой от a до h, а каждая горизонталь - арабской цифрой от 1 до 8. Таким образом каждая клетка шахматной доски обозначается двумя символами – буквой вертикали и цифрой горизонтали, к которым принадлежит эта клетка, например b2 или e4 (представлена на рисунке 5.2).

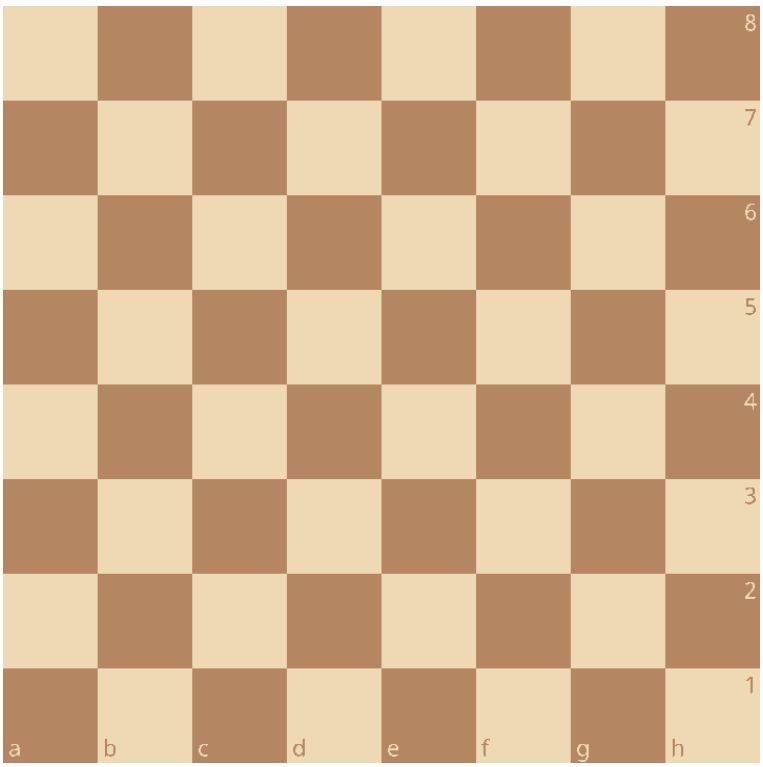


Рисунок 5.2 – Шахматная доска

Шахматная фигура может находится на одной определенной клетке доски. На одной клетке может находится только одна фигура. Для записи положения фигуры будет использоваться стандартное обозначение - имя фигуры и имя клетки, на которой располагается фигура. Фигуры обозначаются так: N - конь, B - слон, R - ладья, Q - ферзь, K - король. Пешка никак не обозначается. Так, например, запись Nb1 означает, что конь находится на клетке b1, запись e8 означает, что пешка находится на клетке e8. Пробел между именем фигуры и именем клетки не ставится.

Правила игры:

Пешки могут двигаться только на одну клетку по вертикали. Белые пешки двигаются «вверх» - в направлении увеличения числа в имени клетки, а черные – «вниз» - в направлении уменьшения. Назад пешки ходить не могут. Исключение - белые пешки, находящиеся на 2 горизонтали, и черные - на 7 горизонтали. Эти пешки могут ходить как на одну клетку вперед, так и на две.

Ладья движется по горизонтали и вертикали на любое доступное количество клеток.

Слон движется по горизонтали на любое доступное количество клеток.

Ферзь может перемещаться на любое количество клеток по горизонтали, вертикали или диагонали.

Конь ходит буквой «Г»: две клетки в одном направлении (горизонтально или вертикально) и затем одну клетку в сторону.

На рисунке 5.3 зеленым кружком отмечены клетки, на которые могут переместиться соответствующие фигуры.

Обратите внимание, что фигуры не могут выходить за пределы доски.

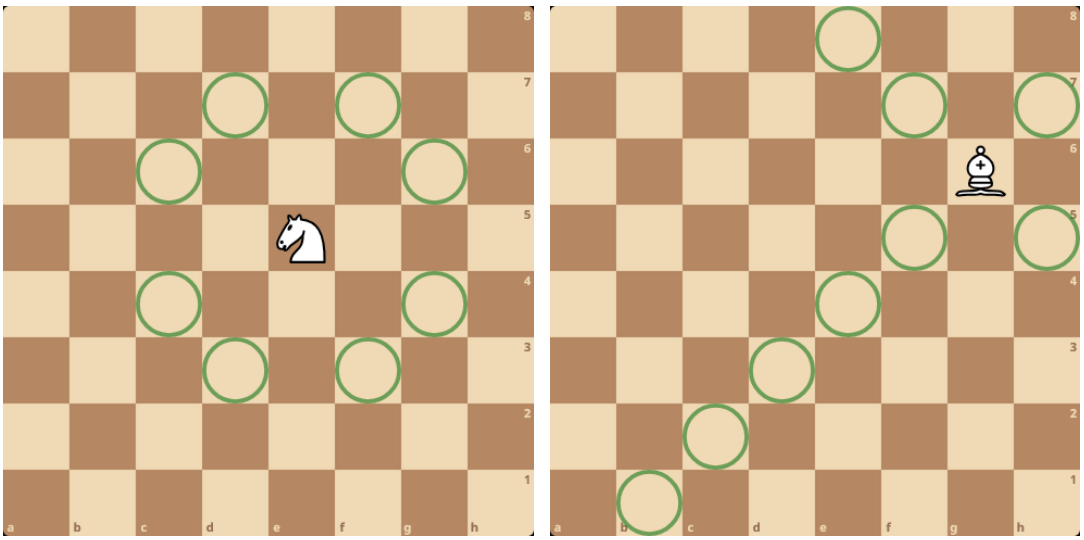


Рисунок 5.3 – Возможные варианты хода фигур «Конь» и «Слон»

Король перемещается на одну клетку в любом направлении (горизонтально, вертикально или по диагонали).

Фигуры не могут двигаться за пределы доски. Фигуры за исключением коня не могут «перепрыгивать» через другие фигуры, в том числе короля.

Шах - это ситуация, когда король находится на клетке под боем фигуры или пешки противоположного цвета. Клетка считается под боем фигуры тогда, когда эта фигура может следующим ходом переместиться со своего текущего положения на эту клетку. Исключение - пешки. Под боем пешки находятся клетки по диагонали.

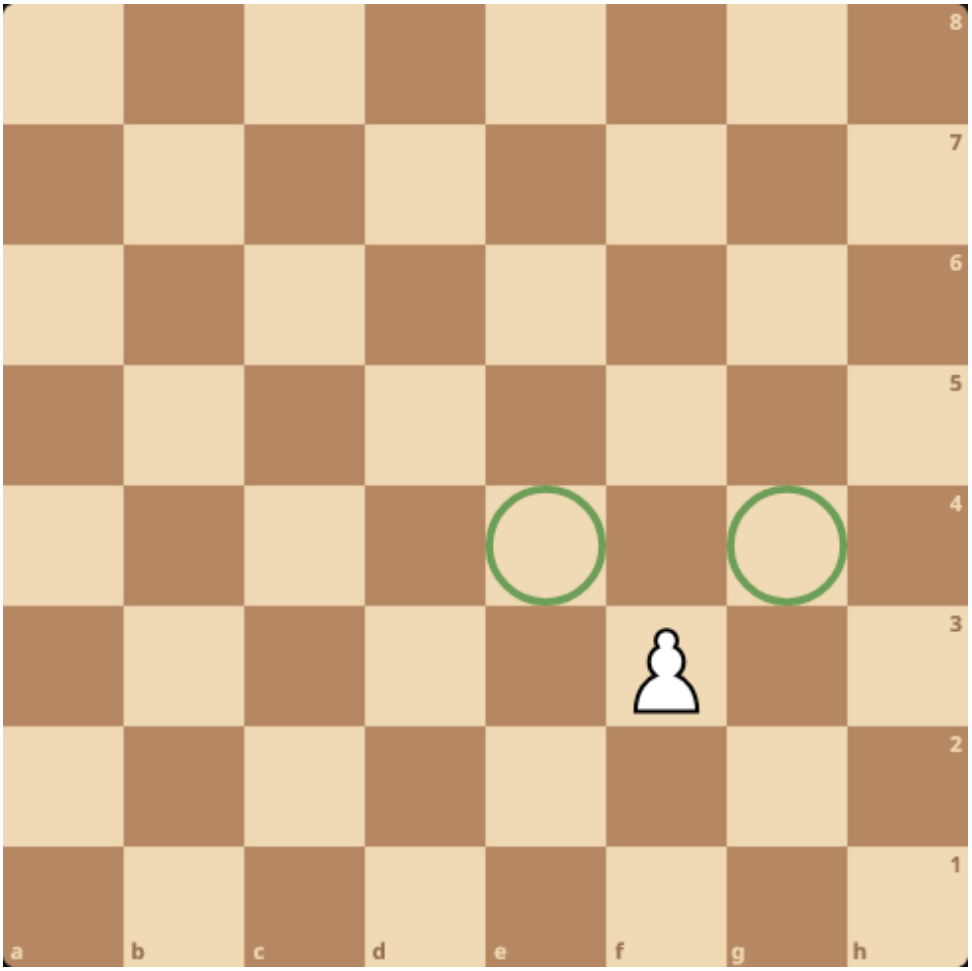


Рисунок 5.4 – Клетки под боем фигуры пешка

На рисунке 5.4 зеленым кружком обозначены клетки, находящиеся под боем пешки. Обратите внимание, что направление боя для пешки зависит от ее цвета и, как следствие, направления движения.

Формат входного файла:

Входной файл состоит из двух строк:

- в первой строке обязательно указывается положения черного короля и как минимум одной из черных фигур, разделенных пробелом;
- во второй строчке указываются положения белых фигур, количество фигур равно одному и более, разделённые пробелом.

Пример файла:
c7 Kc8 Re4
Kf1 Nc5
соответствует такому положению на доске (рисунок 5.5):

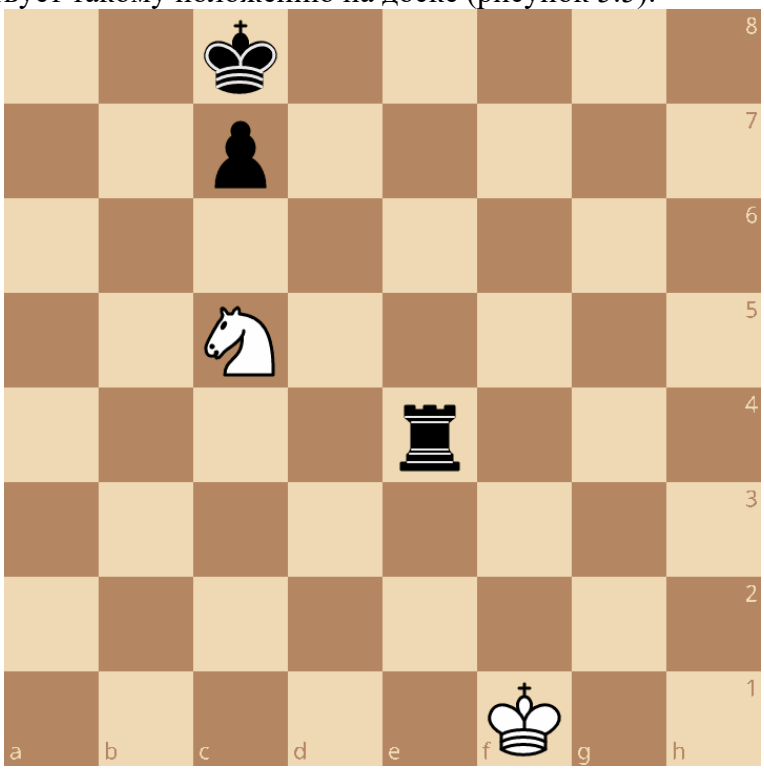


Рисунок 5.5 – Положение фигур

А такой файл:
Rh7 Ka8 Nd7
Ke1 Qe4
соответствует такому положению на доске (рисунок 5.6):

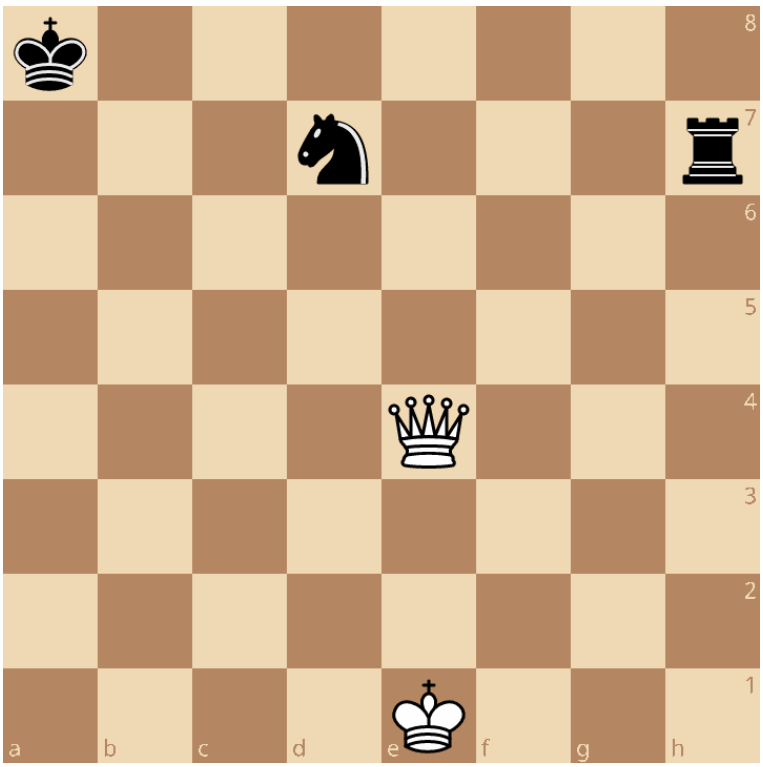


Рисунок 5.6 – Положение фигур

Формат выходного файла:

Если ситуация на шахматной доске определена как «двойной удар с шахом», то программа должна записать в выходной файл белую фигуру, создавшую эту ситуацию, и ее позицию на доске, иначе в выходном файле необходимо разместить значение «0».

Первый пример:
Входной файл:
c7 Kc8 Re4
Kf1 Nc5
Выходной файл должен содержать: 0.

Второй пример:
Входной файл:
Rh7 Ka8 Nd7
Ke1 Qe4
Выходной файл должен содержать: Qe4.