

**ФГОБУ ВО «Финансовый университет при Правительстве
Российской Федерации»
ВСЕРОССИЙСКАЯ ОЛИМПИАДА ПО ИНФОРМАТИКЕ
«МИССИЯ ВЫПОЛНИМА. ТВОЕ ПРИЗВАНИЕ – ФИНАНСИСТ!»
ОЧНЫЙ ЭТАП, 2025 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)

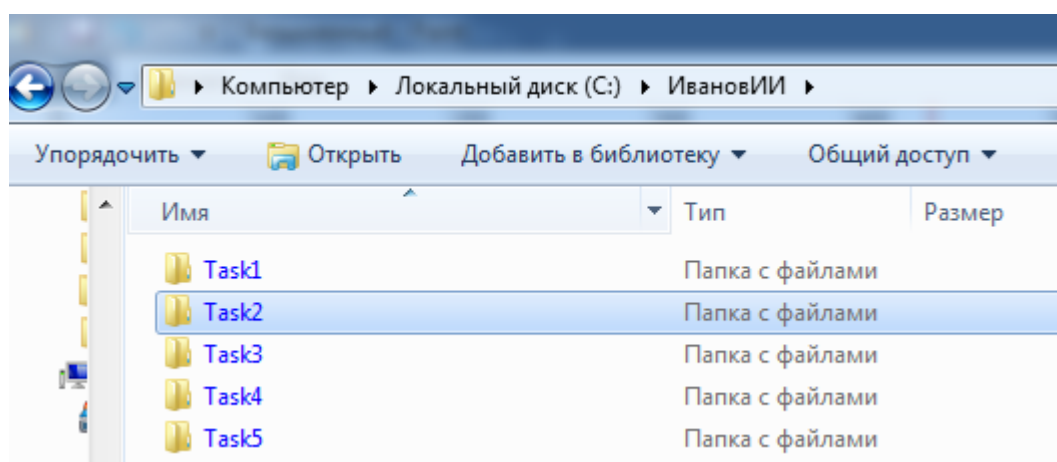


Рисунок П.1 – Структура каталога участника олимпиады

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)

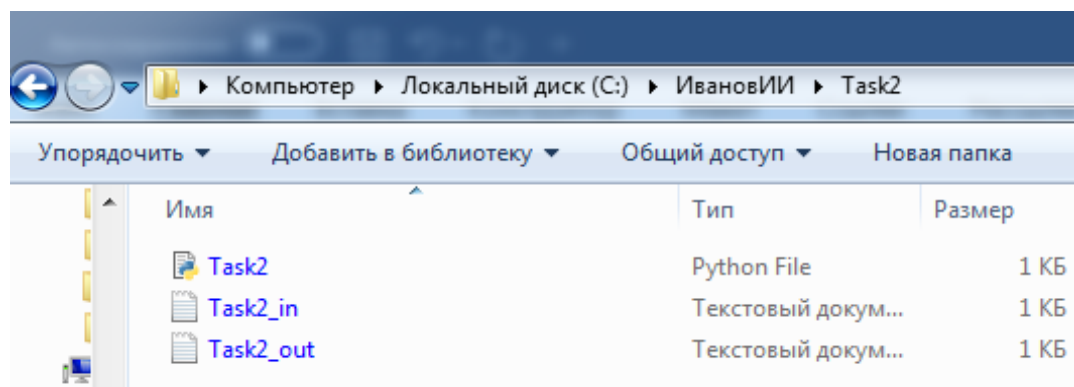


Рисунок П.2 – Размещение файлов в рамках папки задачи

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 комментариев, 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 задании 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 окончании 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).

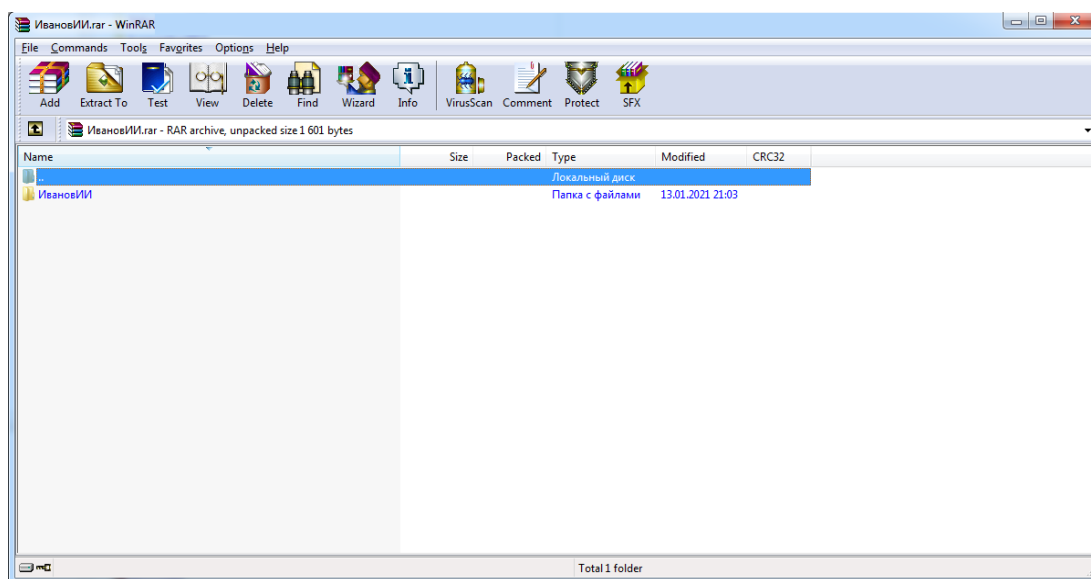


Рисунок П.3 – Пример первого уровня содержимого архива

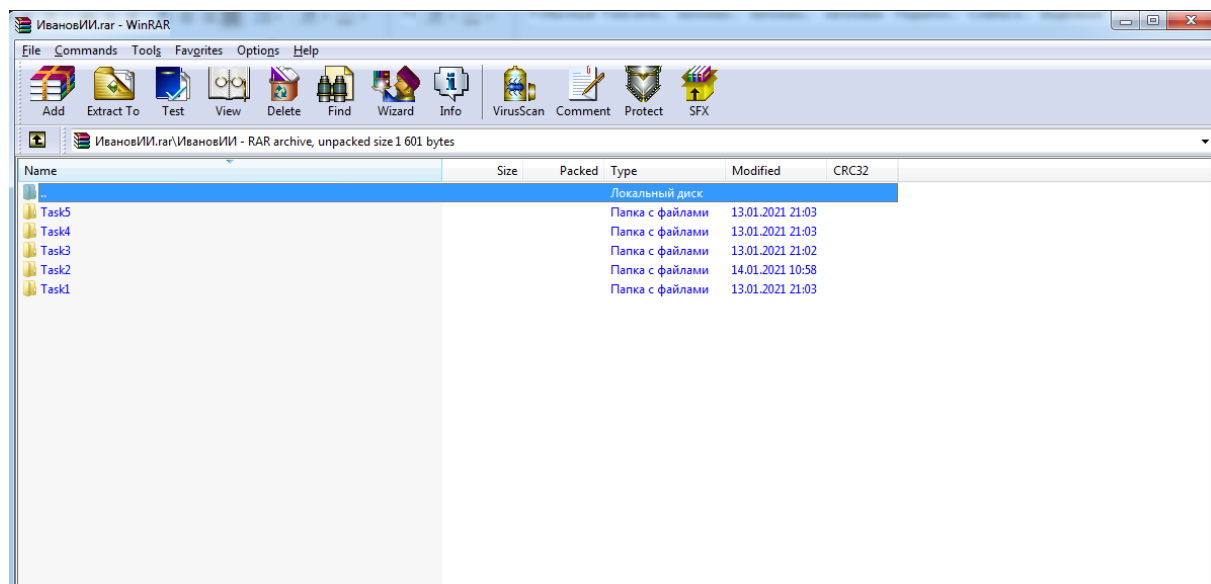


Рисунок П.3 – Пример второго уровня содержимого архива

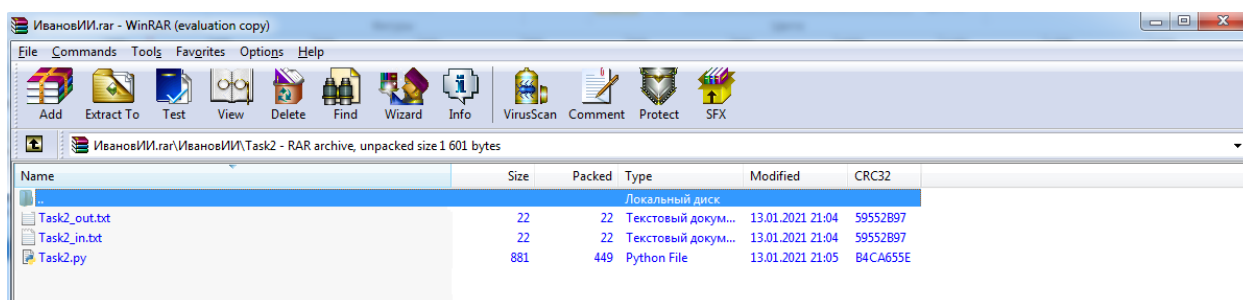


Рисунок П.3 – Пример третьего уровня содержимого архива

Члены комиссии, при проверке заданий, используют операционные системы семейства Windows (версии 7-11). Если члену комиссии не удастся 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.

$$y = \frac{\sqrt{(x-3)}}{\log_2(x^2-16)} + \frac{1}{x+3}$$

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.

$$F = (A \equiv B) \text{ and } (C \text{ or not } D) \text{ or } (A \text{ and } D)$$

Рисунок 2.1 – Логическая функция четырех переменных

В функции обозначения соответствуют:

- *or* – логическое ИЛИ (дизъюнкция);
- *and* – логическое И (конъюнкция);
- *not* – логическое отрицание (НЕ);
- $\rightarrow$ – импликация;
- $\equiv$ – эквивалентность.

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).

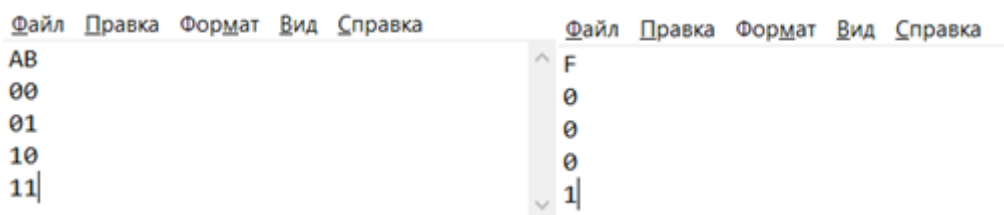
```
Файл  Правка  Формат  Вид  Справка
ABCD
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111|
```

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 а) и 2.3 б) показаны примеры файлов Task2_in.txt и Task2_out.txt,

заполненных в соответствии с требованиями, для логической функции от двух переменных $F = A \text{ and } B$.



а) b)
Рисунок 2.3 – а) Файл Task2_in.txt, б) Файл Task2_out.txt

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

$$F = (A \text{ or } B \text{ and } C) \text{ or } (C \text{ and } D) \text{ or } (\text{not } B \equiv \text{not } C) \leftarrow D$$

Рисунок 2.1 – Логическая функция четырех переменных

В функции обозначения соответствуют:

- *or* – логическое ИЛИ (дизъюнкция);
- *and* – логическое И (конъюнкция);
- *not* – логическое отрицание (НЕ);
- $\leftarrow$ – обратная импликация;
- $\equiv$ – эквивалентность.

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).

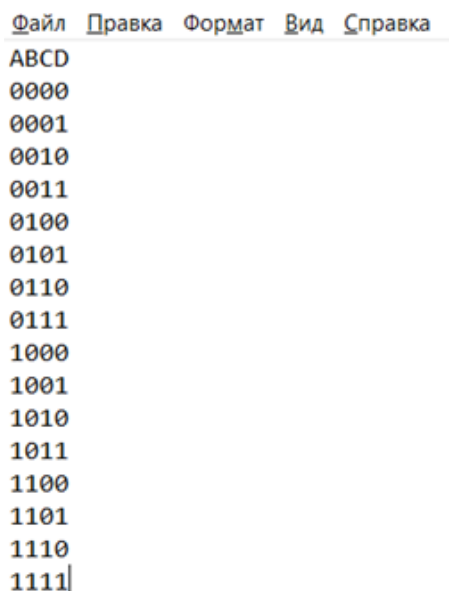


Рисунок 2.2 – Файл Task2_in.txt

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 а) и 2.3 б) показаны примеры файлов Task2_in.txt и Task2_out.txt, заполненных в соответствии с требованиями, для логической функции от двух переменных $F = A \text{ and } B$.

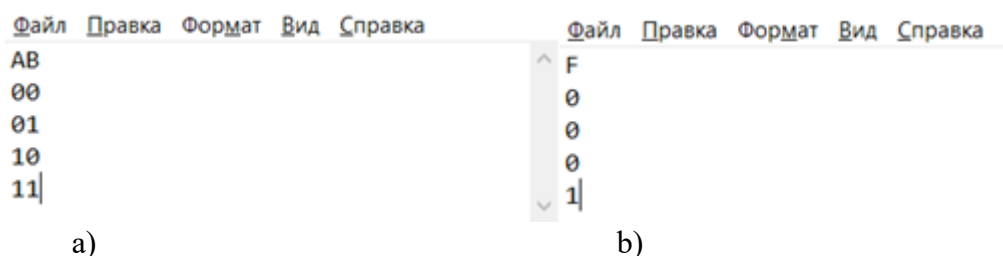


Рисунок 2.3 – а) Файл Task2_in.txt, б) Файл Task2_out.txt

Задание 3 (20 баллов)

В переписке между филиалами Финуниверситета для передачи сообщений решили использовать шифр. Длина сообщений при этом никогда не превышает 20 символов, включая пробелы.

Процесс шифрования начинается с построения сетки размером 6×6. Каждая строка и столбец сетки задаётся одной из 6 букв: «А», «D», «F», «G», «V», и «X». Заполнение сетки происходит в случайном порядке 26 буквами и 10 цифрами, поэтому получатель должен знать точное расположение каждого элемента для успешной дешифровки.

	A	D	F	G	V	X
A	1	J	R	4	H	D
D	E	2	A	V	9	M
F	8	P	I	N	K	Z
G	B	Y	U	F	6	T
V	5	G	X	S	3	O
X	W	L	Q	7	C	0

Рисунок 3.1

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

На первом шаге каждый символ сообщения заменяется на пару букв, обозначающих строку и столбец соответствующего символа в сетке, так А будет заменено на DF, а В — на GA.

Сообщение: АТТАСК.

Шифротекст на первом шаге: DF GX GX DF XV FV

На втором шаге применяется перестановка. Перестановка осуществляется в зависимости от ключевого слова, которое должно быть известно получателю. Пусть в нашем примере таким словом будет «SECRET». Процесс перестановки заключается в следующем. Вначале создаётся новая сетка, в верхней строке которой записываются буквы ключевого слова. Затем под этим словом построчно записывается, полученный на первом шаге зашифрованный текст.

S	E	C	R	E	T
D	F	G	X	G	X
D	F	X	V	F	V

Рисунок 3.2

Затем буквы ключевого слова переставляются в алфавитном порядке вместе с соответствующими им столбцами сетки

C	E	E	R	S	T
G	F	G	X	D	X
X	F	F	V	D	V

Рисунок 3.3

После чего буквы каждого столбца выписываются поочерёдно сверху вниз. Полученная последовательность букв образует окончательный вид шифротекста.

Окончательный вид шифротекста: GXFFGFXVDDXV.

Для восстановления исходного текста необходимо выполнить действия, обратные шифрованию. Обладая ключевым словом, последовательность столбцов можно привести к первоначальному порядку. Зная расположение символов в исходной таблице, можно расшифровать текст.

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.

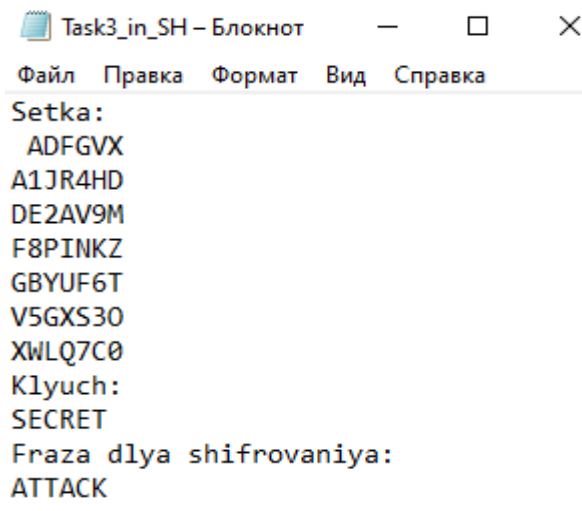


Рисунок 3.5

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

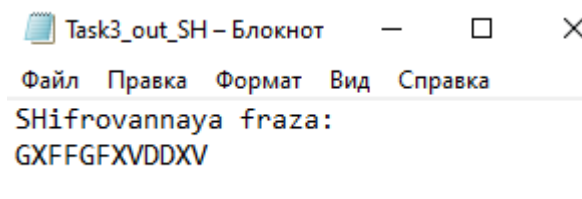


Рисунок 3.6

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

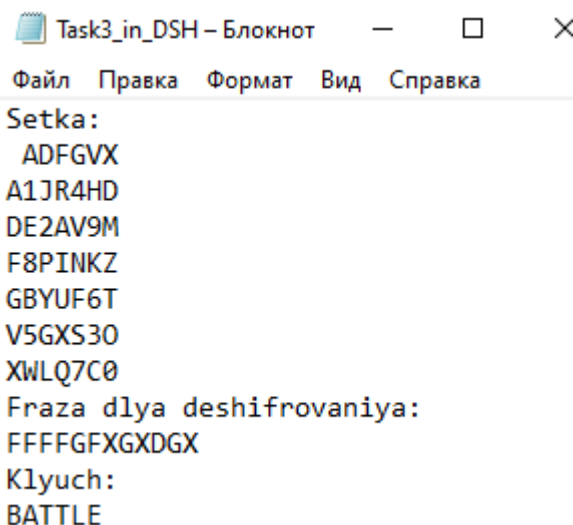


Рисунок 3.7

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

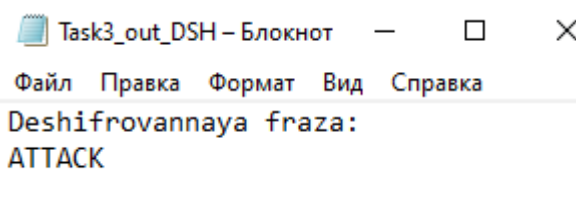


Рисунок 3.8

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 цифры, буквы латинского алфавита и пробелы);
2. Длина сообщения не превышает 20 символов (пробелы из сообщения удаляются или сообщение вводится без пробелов).

При несоответствии требованиям программное средство должно сформировать следующие сообщения 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)

При проведении крупного исследования формируются два набора характеристик для каждой группы объектов. Первый набор (базовый) содержит ключевые параметры каждого объекта в виде прямоугольной матрицы $m \times n$. Второй набор (дополнительный) содержит другие показатели того же объекта, теперь в матрице $n \times l$. Оба набора состоят из k парных матриц, т. е. первая матрица базовая соответствует первой дополнительной, вторая базовая – второй дополнительной и т. д. Нужно собрать эти данные в одну итоговую форму,

перемножив каждую пару матриц, чтобы получить матрицу $m \times l$. Затем сохранить три матрицы (базовую, дополнительную и их произведение) для каждого объекта в единый результирующий файл.

Создайте самостоятельно исходный файл Task4_1_in.txt, который хранит k базовых матриц размерности $m \times n$, где первая строка содержит три натуральных числа k , m , n (k – количество базовых матриц, m и n – размеры каждой из этих матриц), а далее идут k блоков, в каждом по m строк, причём в каждой строке ровно n целых чисел. Создайте также самостоятельно файл Task4_2_in.txt, который хранит k' дополнительных матриц размерности $n' \times l$, в котором первая строка содержит три натуральных числа k' , n' , l , а затем, аналогично, записано k' блоков по n' строк, в каждой строке l целых чисел.

Требуется:

- 1) считать данные из обоих файлов;
- 2) проверить корректность: $k = k'$ (количество пар матриц должно совпадать), $n = n'$ (чтобы матрицы можно было умножать, размерности должны согласовываться, т. е. $(m \times n) \cdot (n \times l) = (m \times l)$); k , m , n , k' , n' , l — натуральные (≥ 1);
- 3) во всех матрицах m или n' строк и n или l столбцов; все элементы – целые числа, если где-то формат нарушен, вывести сообщение об ошибке и завершить;
- 4) для каждой пары матриц (из первого и второго файла под одним номером) выполнить произведение $A_{(m \times n)} \times B_{(n \times l)} = C_{(m \times l)}$;
- 5) все результаты (тройки «базовая матрица – дополнительная матрица – результат») сохранить в третий файл.

Выходные данные – это файл Task4_out.txt, в котором первая строка содержит четыре числа k , m , n , l , затем k «комплектов» из трёх матриц подряд: матрица из Task4_1_in.txt ($m \times n$), соответствующая матрица из Task4_2_in.txt ($n \times l$) и их произведение ($m \times l$).

Если формат не соответствует требованиям ($k \neq k'$, $n \neq n'$, неправильные размеры, нечисловые данные и т. п.), выводится сообщение об ошибке, а файл Task4_out.txt не формируется.

Предположим, что в Task4_1_in.txt записаны две матрицы размером 2×3 , т. е. $k = 2$, $m = 2$, $n = 3$ (рисунок 4.1).

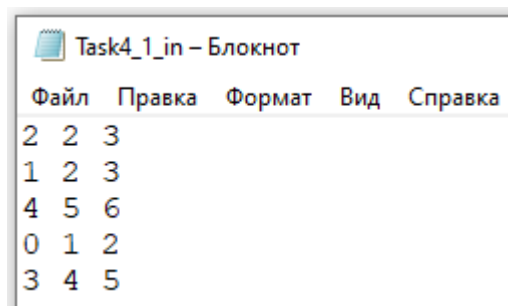


Рисунок 4.1. – Пример файла Task4_1_in.txt.

Здесь первая матрица 2×3 :

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix},$$

вторая матрица 2×3 :

$$\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix}.$$

В Task4_2_in.txt тоже две матрицы размера 3×2 , т. е. $k' = 2$, $n' = 3$, $l = 2$ (рисунок 4.2):

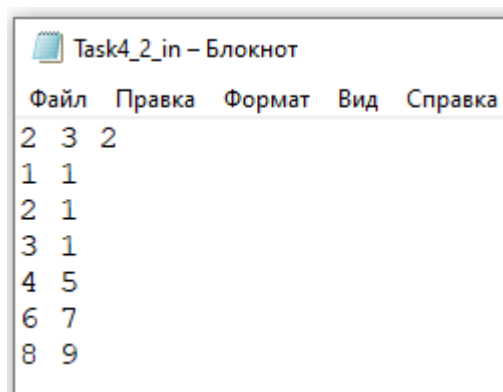


Рисунок 4.2. – Пример файла Task4_2_in.txt.

Здесь первая матрица 3×2 :

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix},$$

вторая матрица 3×2 :

$$\begin{pmatrix} 4 & 5 \\ 6 & 7 \\ 8 & 9 \end{pmatrix}.$$

Проверяем: $k = k' = 2$, $n = n' = 3$. Все сходится, можно умножать. Произведение для первой пары:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 14 & 6 \\ 32 & 15 \end{pmatrix},$$

произведение для второй пары:

$$\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix} \times \begin{pmatrix} 4 & 5 \\ 6 & 7 \\ 8 & 9 \end{pmatrix} = \begin{pmatrix} 22 & 25 \\ 76 & 88 \end{pmatrix}.$$

В итоге файл Task4_out.txt будет хранить следующие данные (рисунок 4.3): первая строка содержит $k=2$, $m=2$, $n=3$, $l=2$, далее идет два «набора» матриц размерностей 2×3 , 3×2 , 2×2 , аналогично для второй пары.

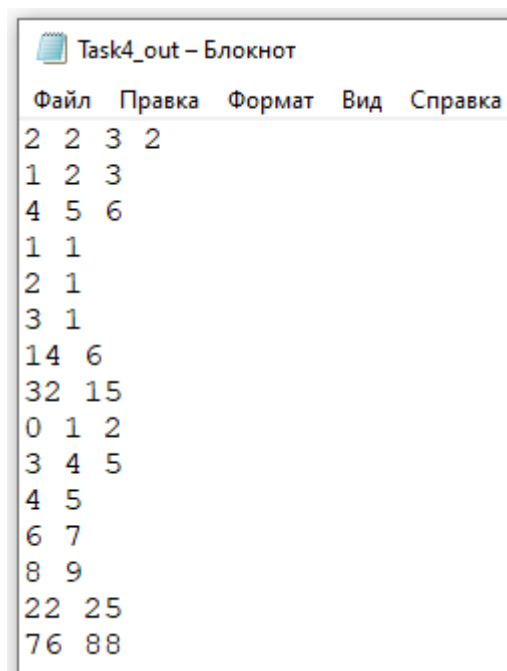


Рисунок 4.3. – Пример файла Task4_out.txt.

После успешной обработки вывести на экран содержимое: Task4_1_in.txt (исходные базовые матрицы), Task4_2_in.txt (дополнительные матрицы), Task4_out.txt (три матрицы подряд по каждому объекту).

Если $k \neq k'$ или $n \neq n'$, или m , n , l не натуральные, или в матрицах не совпадает количество строк/столбцов, – это ошибка (не формируем Task4_out.txt). Если встречаются нечисловые символы или неожиданные пропуски, тоже ошибка. Во всех случаях ошибки вывести сообщение («Ошибка формата данных», «Невозможно перемножить матрицы» и т. п.) и завершить работу.

Следует обратить внимание, что все входные данные (файлы Task4_1_in.txt и Task4_2_in.txt) формируются вручную в соответствии с указанным форматом. Чтобы при отладке программы обеспечить правильную обработку ошибок, некоторые данные могут быть некорректными (неправильное количество строк или символы, не являющиеся числами и т. д.).

Задание 5 (35 баллов)

Написать программу, которая определяет ситуацию МАТ белому королю на шахматной доске. На шахматной доске может находиться любое количество черных и белых фигур (пешка, слон, конь, ладья или ферзь), белый король находится обязательно.

Определение:

МАТ — ситуация в шахматах, когда король находится под шахом, и игрок не может сделать ни одного хода, чтобы его избежать (пример мата на рисунке 5.1). Таким образом, при мате одновременно:

король находится под шахом;

у короля нет возможности уйти от шаха - все поля вокруг него заняты своими фигурами или находятся под ударом фигур противника.

у игрока нет возможности закрыться от шаха другой фигурой.
нет возможности взять фигуру, объявившую шах.

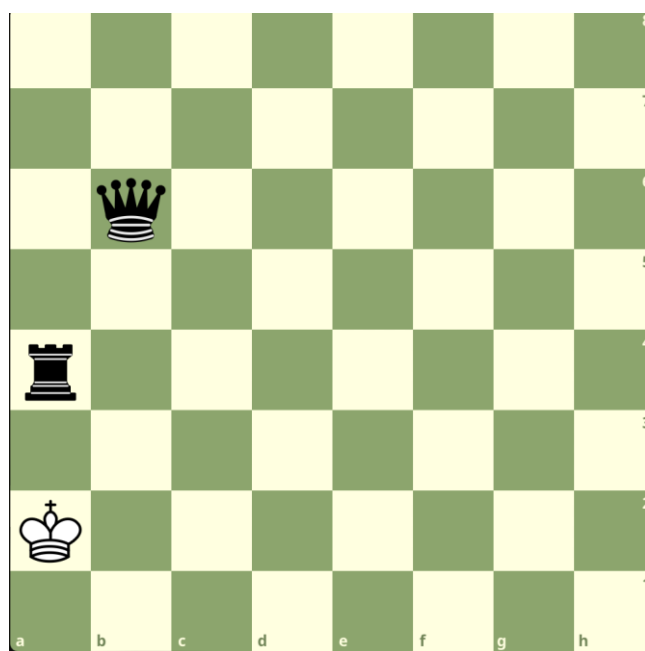


Рисунок 5.1 – Пример ситуации МАТ

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 а до h, а каждая горизонталь - арабской цифрой от 1 до 8. Таким образом каждая клетка имеет свой уникальный адрес, например b2 или e4 (представлена на рисунке 5.2.).

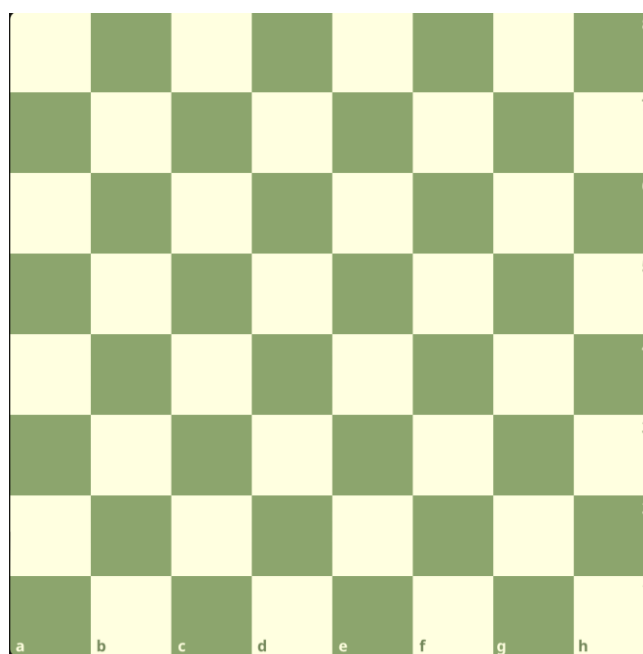


Рисунок 5.2 – Шахматная доска

Шахматная фигура может находиться на одной определенной клетке доски. На одной клетке может находиться только одна фигура. Для записи положения фигур будут использоваться стандартное обозначение - имя фигуры и имя клетки, на которой располагаются фигуры. Фигуры обозначаются так: N - конь, B - слон, R - ладья, Q - ферзь,

К - 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.

Конь движется на две клетки в одном из четырех направлений и на одну клетку в одном из двух, направлений, перпендикулярном первому:

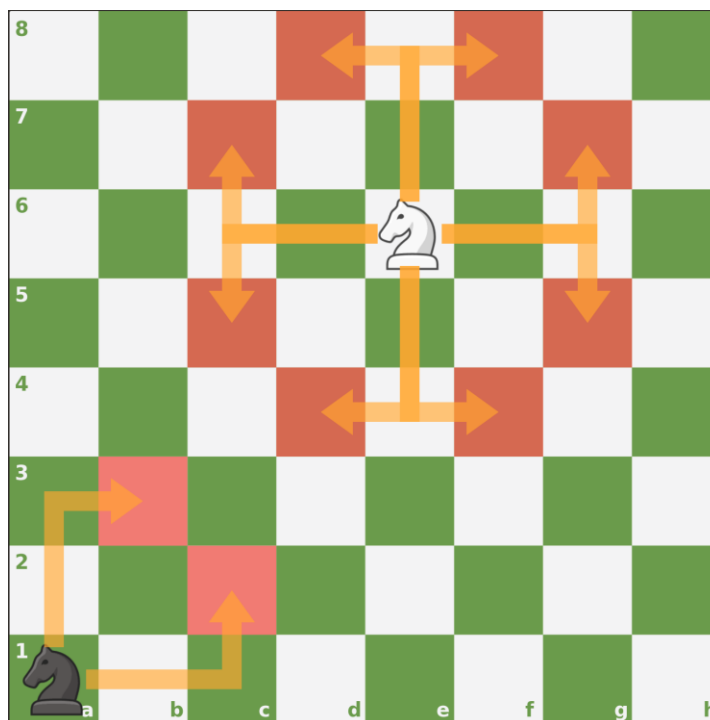


Рисунок 5.3 – Ход фигуры Конь

На рисунке 5.3 красным отмечены клетки, на которые может переместиться соответствующий конь. Конь, расположенный в середине доски может переместиться на любую из восьми клеток. Конь, расположенный в углу может переместиться на любую из двух клеток потому, что фигуры не могут выходить за пределы доски.

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

Слон движется по диагонали на любое доступное количество клеток.

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:

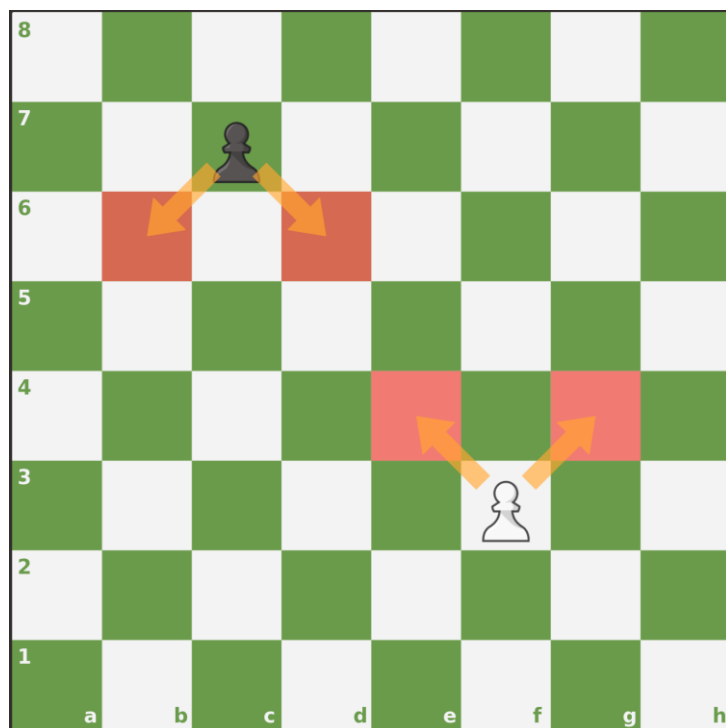


Рисунок 5.4 – Клетки под боем фигуры пешка

На рисунке 5.4 красным обозначены клетки, находящиеся под боем ближайших пешек. 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 указываются положения черных фигуры, количество фигур равно одному и более, разделённые пробелом
- во второй строке обязательно указывается положения белого короля и любое количество белых фигур, разделенных пробелом (допускается указать положение только белого короля);

Пример файла:

Ne2 Qe4

h2 Kh1 Bg3

соответствует такому положению на доске (рисунок 5.5):

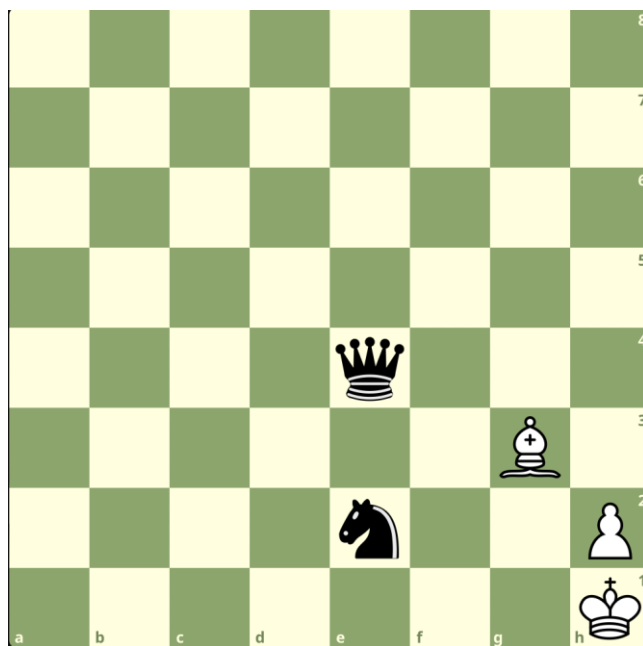


Рисунок 5.5

А такой файл:

Bh6 Rd6

Ke3 Bf4

соответствует такому положению на доске (рисунок 5.6):

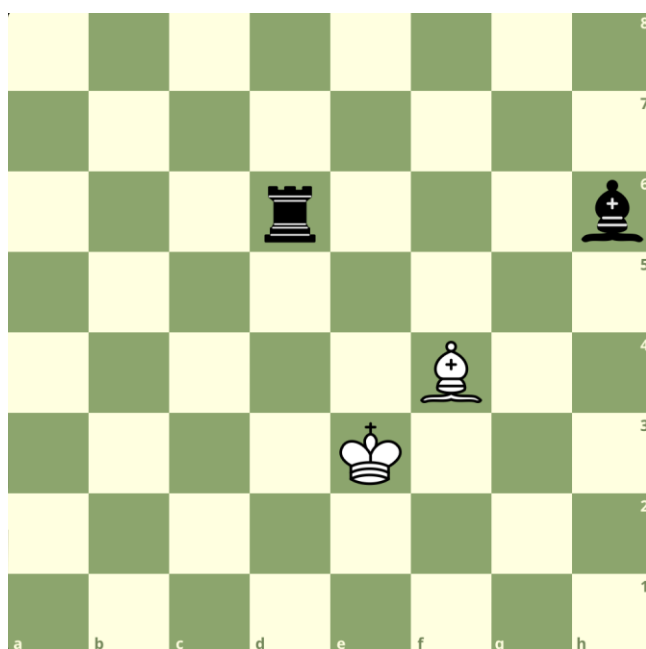


Рисунок 5.6

Формат выходного файла:

Если ситуация на шахматной доске определена как МАТ, то программа должна записать в выходной файл 1, иначе 0.

Первый пример:

Входной файл:

Ne2 Qe4

h2 Kh1 Bg3

Выходной файл должен содержать 1.

Второй пример:

Входной файл:

Bh6 Rd6

Ke3 Bf4

Выходной файл должен содержать 0.