ФГОБУ ВО «Финансовый университет при Правительстве Российской Федерации»
ВСЕРОССИЙСКАЯ ОЛИМПИАДА ПО ИНФОРМАТИКЕ
«МИССИЯ ВЫПОЛНИМА. ТВОЕ ПРИЗВАНИЕ – ФИНАНСИСТ!»
ОЧНЫЙ ЭТАП, 2021 год

Инструкция участнику олимпиады

Продолжительность олимпиады – 240 минут (4 астрономических часа). 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**)**

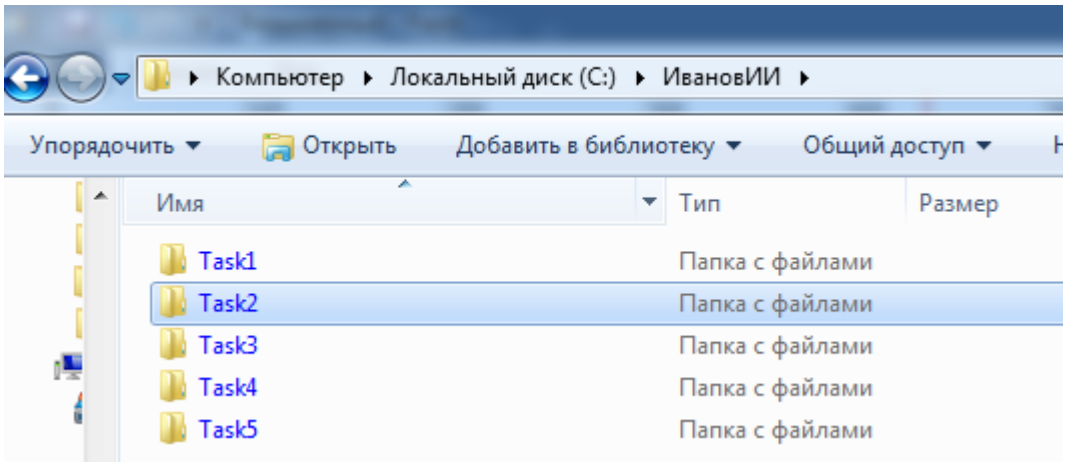


Рисунок П.1 – Структура каталога участника олимпиады

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)

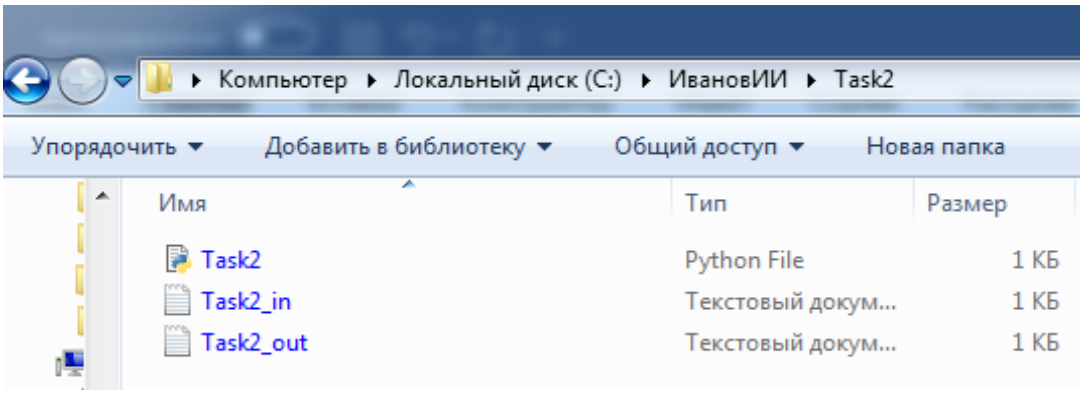


Рисунок П.2 – Размещение файлов в рамках папки задачи

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

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.

По окончанию работы над заданиями участник формирует архив с содержимым решений в соответствии с предложенной выше структурой и размещает его в информационной системе Финансового Университета при Правительстве РФ. Расширение архивного файла должно быть rar или zip (пример см. рис П.3-П.5).

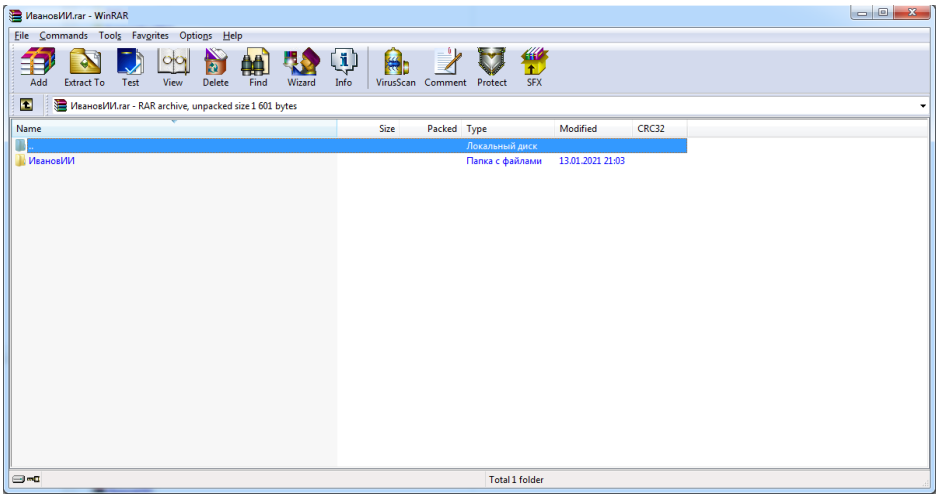


Рисунок П.3 – Пример первого уровня содержимого архива

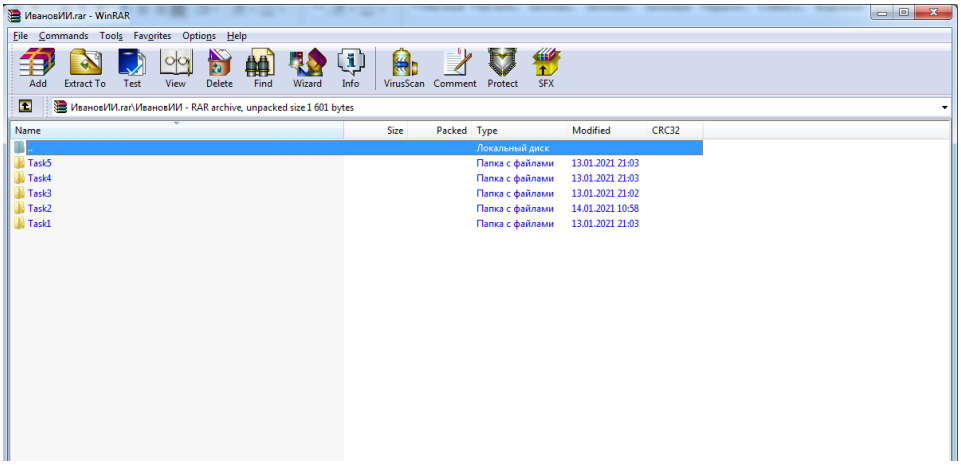


Рисунок П.3 – Пример второго уровня содержимого архива

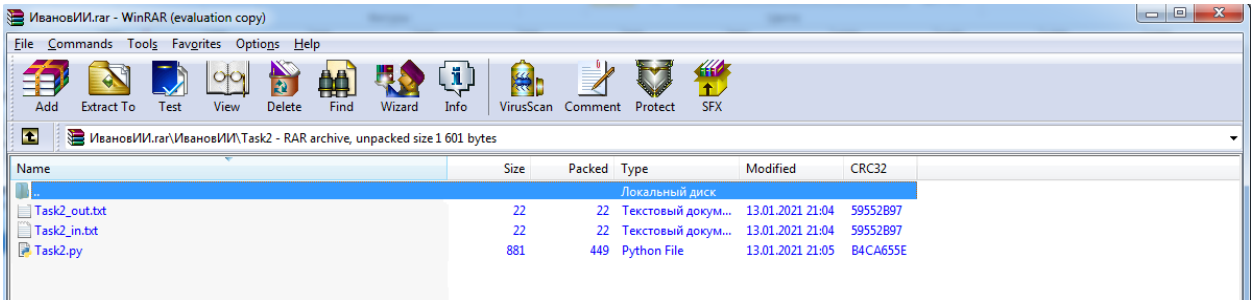


Рисунок П.3 – Пример третьего уровня содержимого архива

Члены комиссии, при проверке заданий, используют операционные системы семейства Windows (версии 7-10). 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.

Олимпиадные задания по информатике
Вариант 2

Задача 1 (8 баллов)

Ребенок пытается вручную определить трехзначные числа кратные 6, и у которых сумма цифр сотни, десятки, единицы в 11 раз меньше самого числа. Помогите ему написав соответствующую программу, которая выведет на экран все трехзначные числа, удовлетворяющие указанному выше критерию. Если таких чисел нет, то выведите на экран сообщение «No».

Задача 2 (12 баллов)

Требуется реализовать программу для определения типа треугольника: остроугольный, прямоугольный или тупоугольный. Исходные данные, в виде длины сторон треугольника, программа берет из созданного текстового файла с именем Task2_in.txt (создать самостоятельно с помощью Блокнота в соответствии с шаблоном на рисунке 2.1). В соответствии с шаблоном, значения длины сторон должны быть разделены «;», целая часть числа отделяется от дробной «.», конец строки обозначается «!».

Программная реализация должна включать функцию, которая принимает в качестве параметров три стороны треугольника и определяет тип треугольника. Если треугольник с заданными сторонами не существует, то программа должна вернуть строку

«No». Полученные результат должно быть записаны в файл Task2_out.txt (пример см. рис. 2.2) один из четырех вариантов: «Ostrougol’nyj», «Prymougol’nyj», «Tupougol’nyj» или «No» (пример файла Task2_out.txt для несуществующего треугольника см. рис. 2.3).

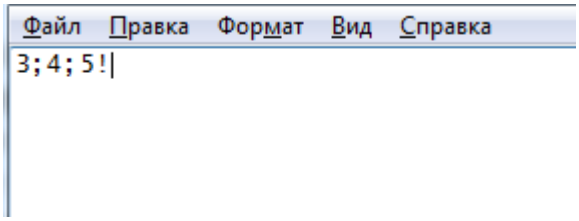


Рисунок 2.1 – Пример текстового файла Task2_in.txt с исходными значениями, соответствующего шаблону

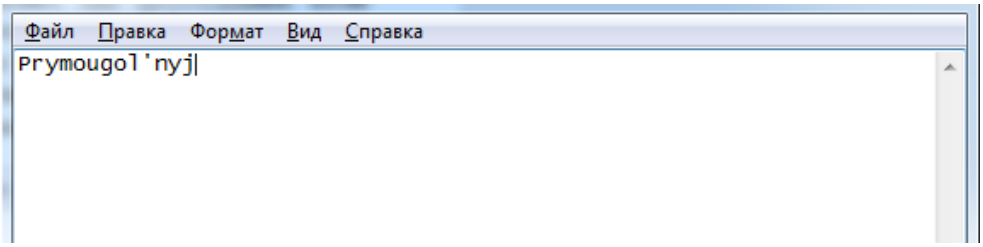


Рисунок 2.2 – Пример текстового файла Task2_out.txt для прямоугольного треугольника

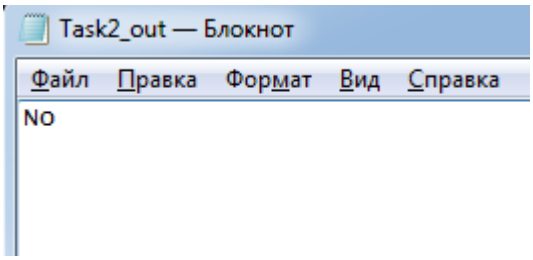


Рисунок 2.3 – Пример текстового файла Task2_out.txt для несуществующего треугольника

Задача 3 (15 баллов)

Выполнить программную реализацию расчета значений логической функции четырех переменных (см. рис. 3.1) и вывод результата в файл Task3_out.txt.

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

Рисунок 3.1 – Логическая функция четырех переменных

В функции обозначения соответствуют:

- *or* – логическое ИЛИ (дизъюнкция);
- and – логическое И (конъюнкция);
- *not* – логическое отрицание (НЕ);
- ← – обратная импликация;
- ≡ – эквивалентность.

Исходные данные требуется считать из файла Task3_in.txt строго в указанном порядке. (пример см. рис. 3.2)

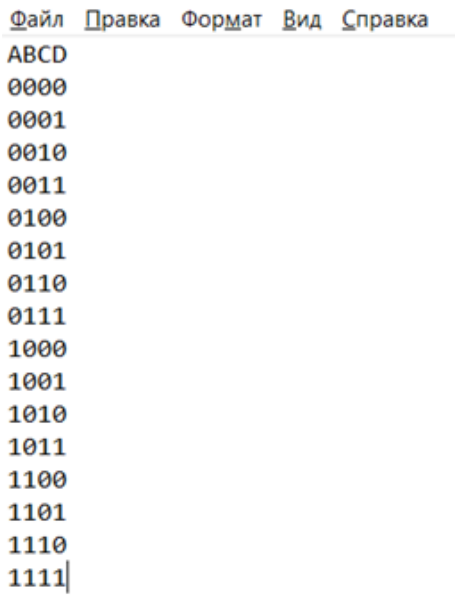
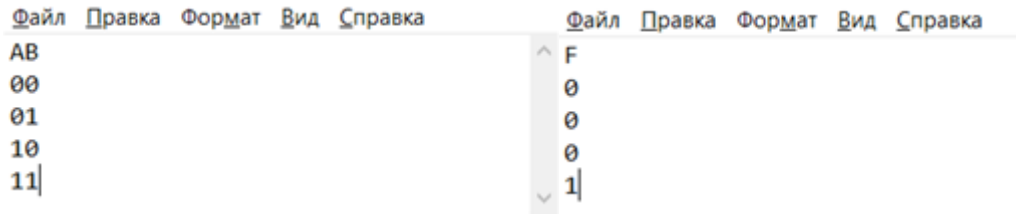


Рисунок 3.2 – Файл Task3_in.txt

Значение F вычисляется разработанной программой (за ручное решение с записью в файл Task3_out.txt баллы не начисляются) и записывается в файл Task3_out.txt.

На рисунках 3.3 a) и 3.3 b) показаны примеры файлов Task3_in.txt и Task3_out.txt, заполненных в соответствии с требованиями, для логической функции от двух переменных F = A and B.



a) b)

Рисунок 3.3 – a) Файл Task3_in.txt, b) Файл Task3_out.txt

Задача 4 (30 баллов)

Работник офиса получил долгожданный отпуск и начал его планирование. В качестве транспорта для поездки в курортный город менеджер выбрал железную дорогу. В планах отпускника составить маршрут, который позволит добраться из Белгорода (начальный пункт) в Кисловодск (конечный пункт). С учетом того, что прямого железнодорожного сообщения между данными городами нет, то потребуется совершить пересадку. Путешественник ограничил себя не более чем двумя пересадками.

Для составления маршрута следования, отпускник выписал расписание всех поездов, с помощью которых он может добраться из Белгорода в Кисловодск через промежуточные пересадки и записал их в текстовый файл Task4_in.txt (см. рис. 4.1).

Формат строки в файле XXX; Город1; Город2; дата отправления; время отправления; дата прибытия; время прибытия,

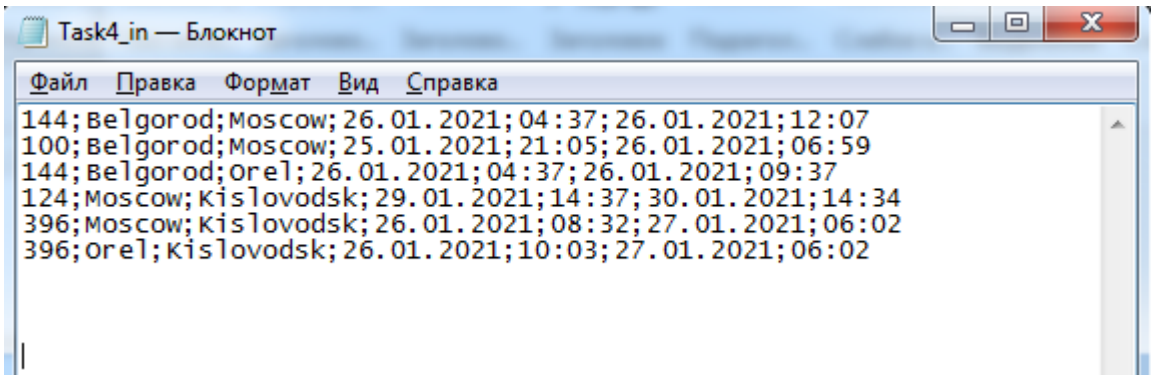


Рисунок 4.1 – Пример расписания поездов

где:

- XXX - номер поезда состоящий из трех цифры;
- Город1 – пункт отправления поезда;
- Город 2 – пункт, в который прибывает поезд;
- DD.MM.YYYY – дата отправления (DD – день, MM – месяц, YYYY – год);
- Время отправления – hh:min (hh – часы, min - минуты);
- Дата прибытия – DD.MM.YYYY;
- Время прибытия – hh:min;
- Все данные в строке разделены «;».

Ограничения по количеству строк во входном файле – 15.

С целью помощи офисному работнику, вам требуется разработать программу, которая составит маршрут следования с наименьшим временем в дороге (время в дороге считается: от времени отправления первого поезда, до прибытия отдыхающего на вокзал в конечном пункте назначения) с учетом расчетов времени в дороге и времени ожидания при пересадках.

Результат вычисления записывается в файл Task_out.txt, пример показан на рисунке 4.2.

Первая строка выходного файла содержит:

- Дата отправления из первого города (Date out): DD.MM.YYYY (DD – день, MM – месяц, YYYY – год);
- Время отправления из первого города (Time out): hh:min (hh – часы, min - минуты);
- XXX;Город1;Город2 (XXX - номер поезда состоящий из трех цифры;Город1 – пункт отправления поезда; Город 2 – пункт, в который прибывает поезд). Количество строк соответствует количеству поездов, на которых требуется перемещаться из начального пункта в конечный;
- Дата прибытия в конечный пункт (Date in)): DD.MM.YYYY (DD – день, MM – месяц, YYYY – год);
- Время прибытия в конечный пункт (Time in): hh:min (hh – часы, min - минуты);
- Время ожидания 1 (Time wait1): dd day; hh hour; mm min (dd – количество полных суток ожидания первой пересадки, hh – количество полных часов ожидания первой пересадки, mm – количество полных минут ожидания первой пересадки)
- Время ожидания 2 (Time wait2): dd day; hh hour; mm min (dd – количество полных суток ожидания второй пересадки, hh – количество полных часов ожидания второй пересадки, mm – количество полных минут ожидания второй пересадки)
- Полное время ожидания пересадок в дороге (All Time wait): dd day; hh hour; mm min (dd – количество полных суток ожидания всех пересадок, hh – количество полных часов ожидания всех пересадок, mm – количество полных минут ожидания всех пересадок);

- Полное время в дороге (All Time way): dd day; hh hour; mm min (dd – количество полных суток в дороге, hh – количество полных часов в дороге, mm – количество полных минут в дороге).

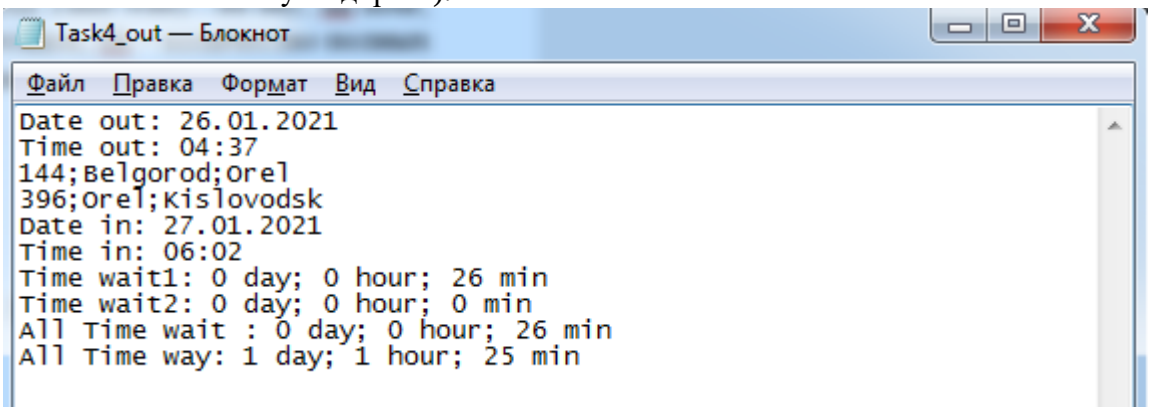


Рисунок 4.2 – Пример выходного файла

При использовании стандартных библиотек для работы с датой и временем снимается 15 баллов.

Задача 5 (35 баллов)

Русские шашки один из вариантов игры в шашки.

В задаче требуется разработать программу, которая позволяет вычислить максимальное количество взятий, которое может совершить игрок в соответствии с правилами русских шашек (см. правила ниже).

Для программной реализации алгоритма вычисления требуется представить поле для игры в текстовом файле. На рисунке 5.1 показано кодирование игрового поля:

- «+» - черное поле;
- «*» - белое поле.

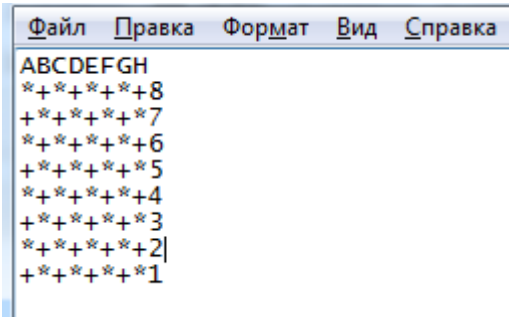


Рисунок 5.1 – Кодирование игрового поля.

В шаблоне кодирования шашек на поле используются следующие обозначения:

- обычная белая шашка изображается символом «l»;
- дамка белого цвета изображается «L»;
- обычная черная шашка изображается символом «b»;
- дамка черного цвета «B».

Пример закодированного расположения шашек на доске с помощью текстового файла Task5_in.txt смотрите на рисунке 5.2

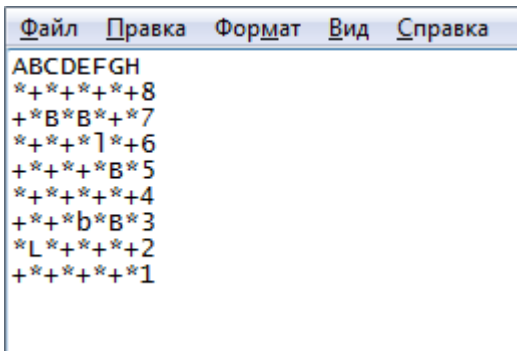


Рисунок 5.2 – Пример расположения шашек на игровом поле в файле Task5_in.txt

Запись вариантов хода осуществляется в соответствии со следующими правилами:

- для записи хода шашки обозначают сначала поле, на котором шашка стояла, затем ставят тире и записывают поле, на которое она ставится (пример d3-e4).
- при записи взятия (боя) вместо тире ставится двоеточие (пример d5:b7);
- при взятии одним ходом нескольких шашек запись хода производится следующим образом: сначала записывается поле, с которого шашка начала свой ход, затем ставится двоеточие и обозначается поле, на которое она встала после боя (пример g8:c4).
- если необходимо отметить направление взятия, то после записи поля, с которого начался бой, последовательно записывают обозначение полей, на которых совершалось изменение направления. Между обозначениями отдельных полей ставится двоеточие (пример d3:f1:h3).

Задача программной реализации состоит в следующем:

- из заданного расположения шашек в файле Task5_in.txt (пример см. рис. 5.3), требуется найти ход для максимального взятия шашек соперника в соответствии с правилами русских шашек;

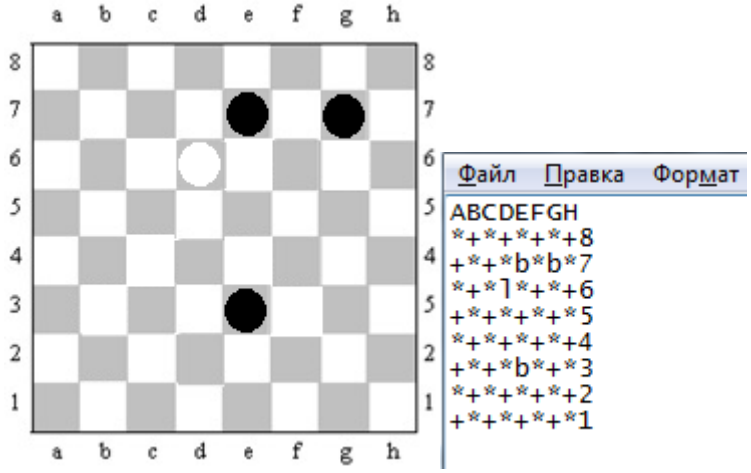


Рисунок 5.3 - Пример отображения шашек на доске и в файле Task5_in.txt

- право хода принадлежит белым шашкам;
- ход с максимальным взятием требуется записать в файл Task5_out.txt (см. рис. 5.4);

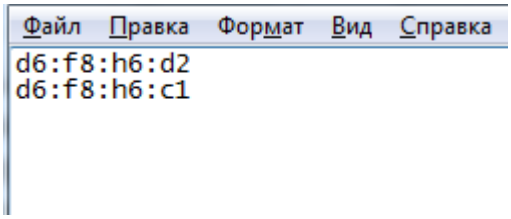


Рисунок 5.4 – Пример выходного файла Task5_out.txt (право хода у белых шашек)

- если количество ходов при взятии максимального количества шашек соперника более одного (например, конечная клетка дамки может быть разной или существует более одного варианта взятия одинакового количества шашек), и они могут быть осуществлены, т. е. у игрока есть выбор (см. рис 5.4), то они записываются в столбик.
- если взятие невозможно, то требуется осуществить запись «No» в файл Task5_out.txt.

Правила игры:

Цель игры - уничтожить все шашки противника или лишить их возможности хода («запереть»). Игра проводится на доске 8×8 клеток. Доска располагается между партнерами таким образом, чтобы слева от играющего находилось тёмное угловое поле. В начальной позиции у каждого игрока по 12 шашек, расположенных в первых трёх рядах на черных клетках.

Первый ход делают белые шашки.

«Простые» шашки могут ходить по диагонали на одно поле вперёд и бить как вперед, так и назад по диагонали, с переходом на пустое поле за битой шашкой.

При достижении последнего противоположного начальной позиции горизонтального ряда простая шашка превращается в дамку.

Дамка ходит на любое количество клеток вперёд и назад по диагонали.

Взятие шашки соперника является обязательным.

При нескольких вариантах взятия можно выбрать любой.

Дамка бьет по диагонали, как вперед, так и назад и становится на любое свободное поле после побитой шашки.

Если простая шашка достигла последнего ряда во время взятия, то она превращается в дамку и может продолжить взятие шашек соперника по правилам дамки.