ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОМСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информатика и вычислительная техника»

Отчёт по лабораторной работе № 4

по дисциплине

«Проектирование и тестирование программного обеспечения»

Выполнил:

Студент гр. ПИН-222

Лубенский И. В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подп., дата)

Проверил:

Старший преподаватель каф. ИВТ

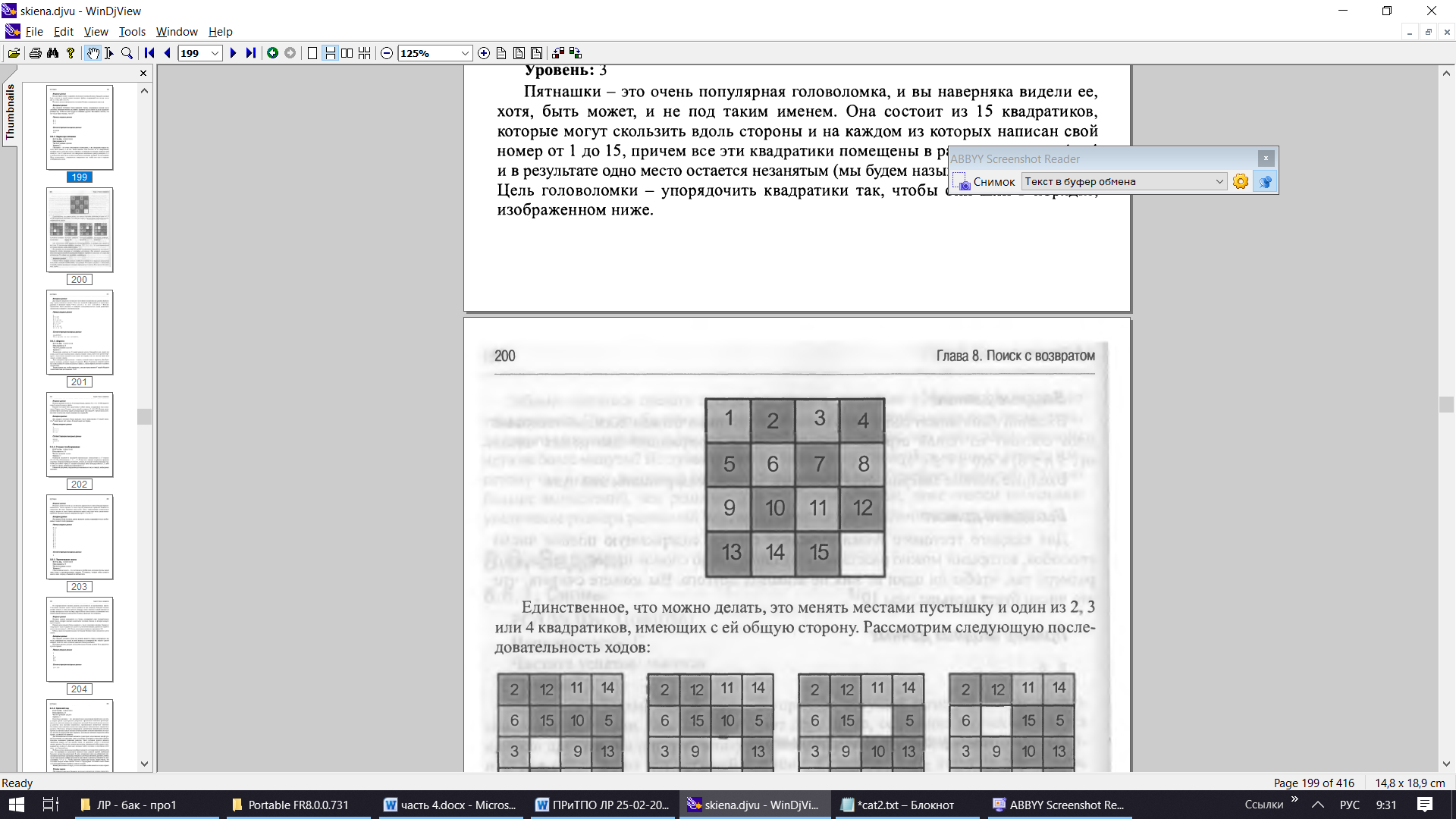
Блохин А. В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подп., дата)

Омск, 2024

**ОПИСАНИЕ ЗАДАЧИ “ПЯТНАШКИ”**

Пятнашки — это очень популярная головоломка, и вы наверняка видели ее. Она состоит из 15 квадратиков, которые могут скользить вдоль стороны и на каждом из которых написан свой номер от 1 до 15, причем все эти квадратики помещены в рамку размером 4x4 и в результате одно место остается незанятым (мы будем называть его пустышкой). Цель головоломки - упорядочить квадратики так, чтобы они шли в порядке, изображенном ниже.



Единственное, что можно делать, это менять местами пустышку и один из 2, 3 или 4 квадратиков, имеющих с ей общую сторону.

Мы обозначаем ходы исходя из соседа пустышки, с которым она меняется местами. Разрешенными являются значения "R", "L", "U", "D" для перемещений пустышки вправо, влево, вверх и вниз.

Вы должны по начальному состоянию головоломки определить последовательность шагов, ведущую к итоговому состоянию. Вы ограничены 50 ходами для решения головоломки (достаточное ли это число для решения любого начального состояния?

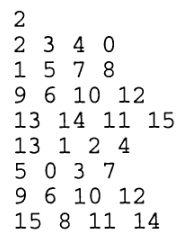
**Входные данные**

Первая строка входных данных содержит целое число и, задающее количество начальных позиций головоломки. Следующие An строк содержат п начальных позиций, причем на каждую позицию приходится 4 строки. Пустышка обозначается нулем.

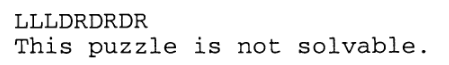
**Выходные данные**

Для каждого введенного начального состояния головоломки вы должны вывести одну строку выходных данных. Если для заданной конфигурации не существует решения, то выведите строку "This puzzle is not solvable". Если же головоломка имеет решение, то выведите последовательность ходов, решающих головоломку в формате, описанном выше.

**Пример входных данных**



**Соответствующие выходные данные**



**АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ “ПЯТНАШКИ”**

Согласно Википедии: “Любую из 10 461 394 944 000 разрешимых конфигураций «классической» головоломки 4 × 4 можно перевести в начальную не более чем за 80 ходов, если под ходом понимать перемещение одной плитки, или не более чем за 43 хода, если под ходом понимать одновременное перемещение сплошного ряда из не более чем трёх плиток. Только 17 из всех разрешимых конфигураций не могут быть решены менее чем за 80 ходов, то есть требуют 80 перемещений отдельных плиток для решения; только 16 разрешимых конфигураций требуют 43 перемещений сплошных рядов плиток.”. Таким образом, большая часть всех возможных разрешимых изначальных конфигураций игры может быть разрешена менее, чем за 50 ходов, поэтому ограничение, поставленное на решение задачи, будет значительным только в очень маленьком количестве случаев.

Схема алгоритма решения поставленной задачи показана на рисунке 1.

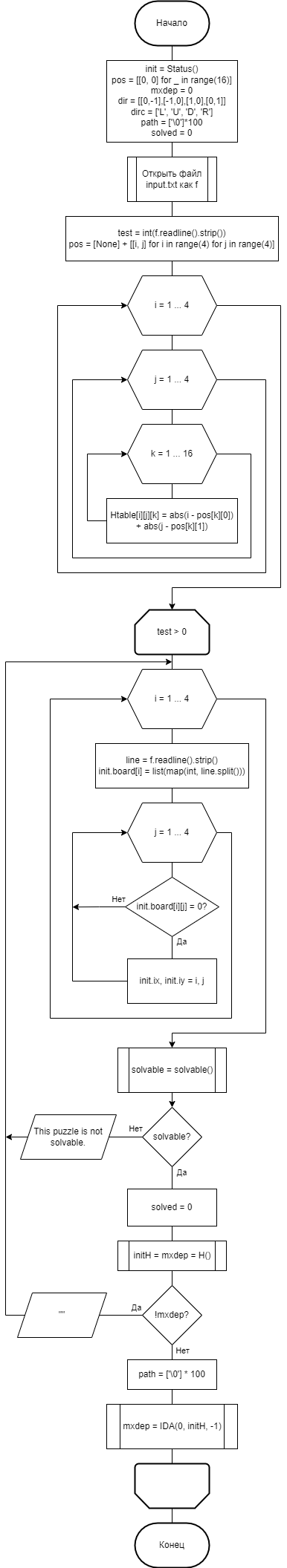


Рисунок 1 — Схема составленного алгоритма

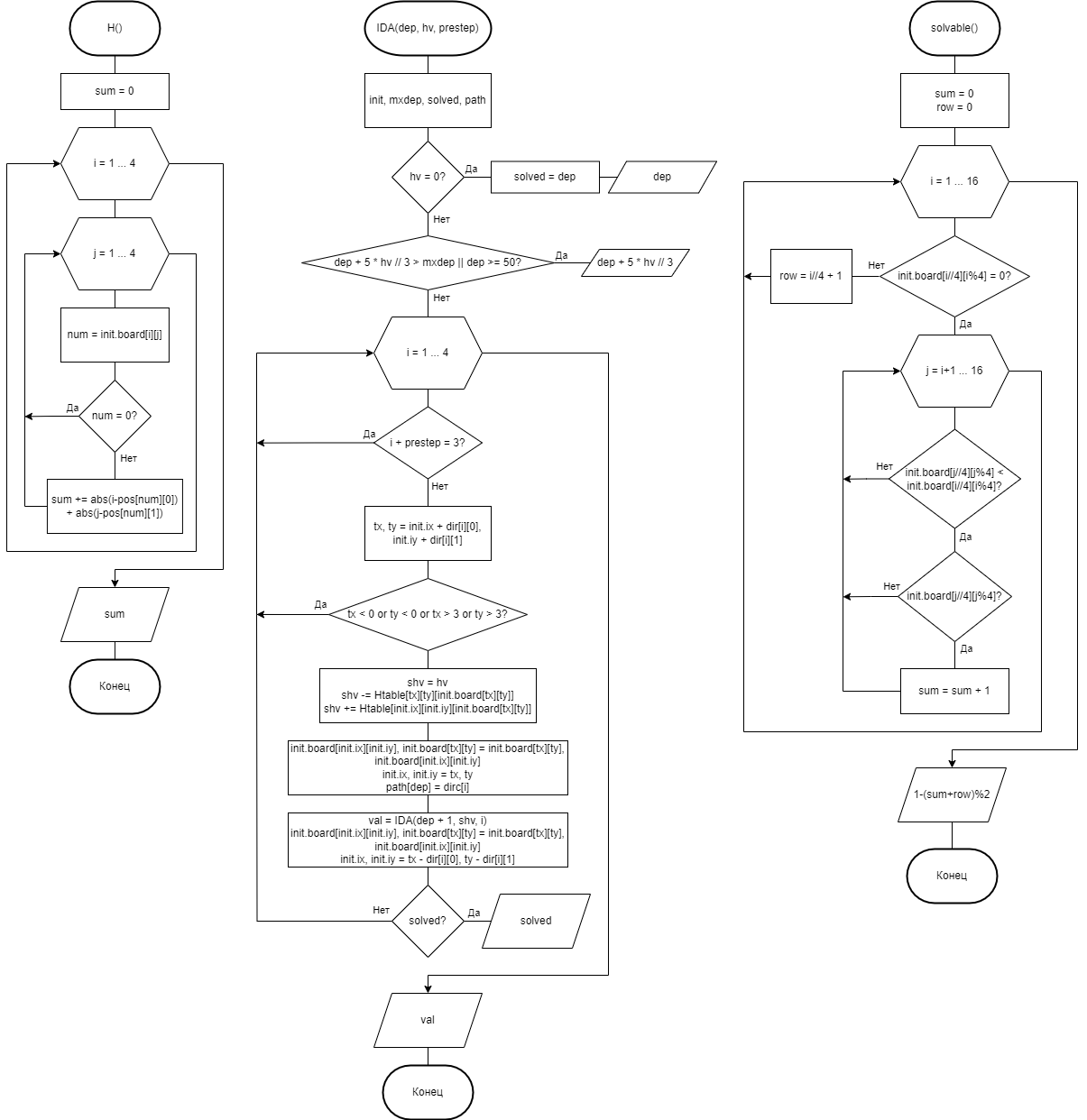


Рисунок 1.1 — Схема составленного алгоритма

Исходный код программы показан на рисунке 2.

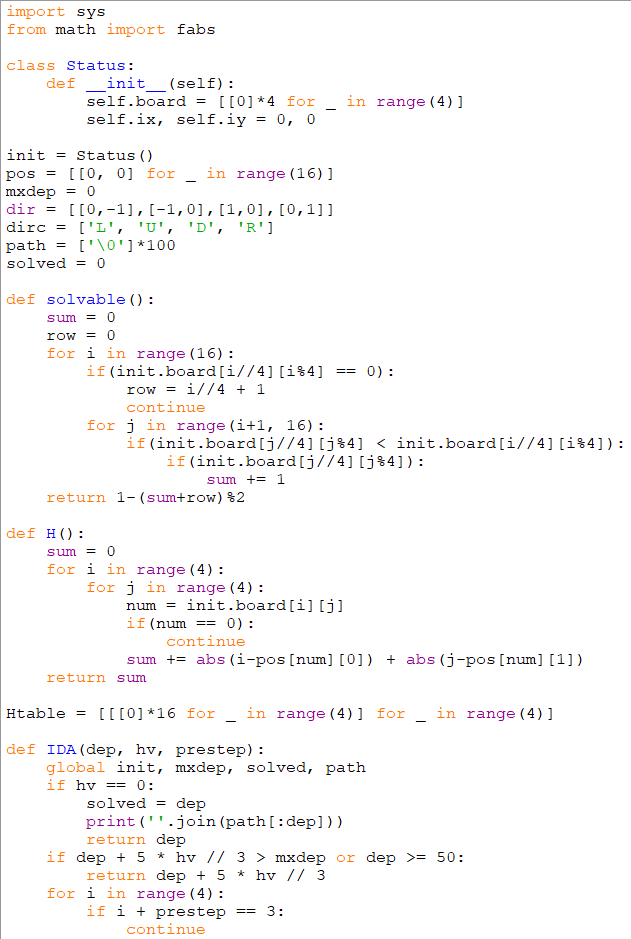


Рисунок 2 — Исходный код для решения задачи

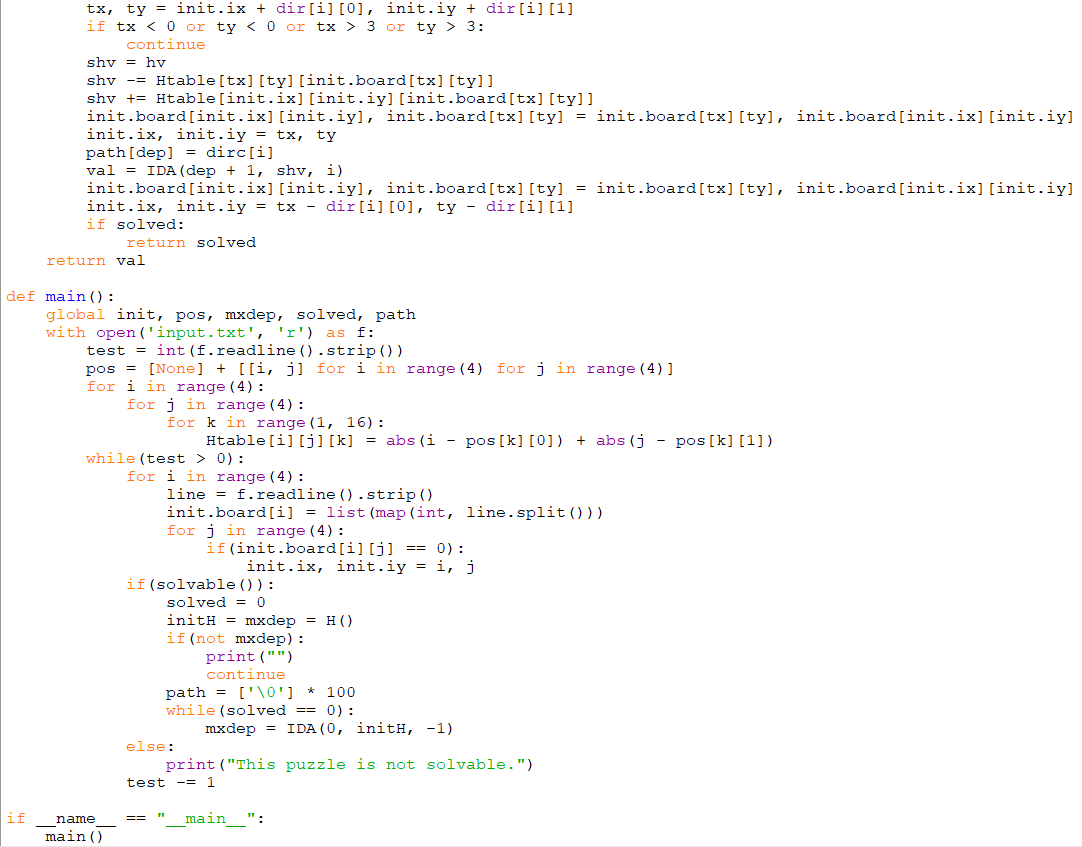


Рисунок 2.1 — Исходный код для решения задачи

В коде определён класс Status для представления состояния игрового поля. Он включает в себя само поле (сетку 4x4) и координаты пустой клетки. Переменная init является экземпляром Status, представляющим начальное состояние головоломки.

Функция solvable проверяет, можно ли решить данную конфигурацию головоломки. Она вычисляет сумму инверсий и номер строки пустой клетки. Головоломка считается решаемой, если сумма инверсий плюс номер строки четны.

Функция H вычисляет эвристику по манхэттенскому расстоянию для головоломки. Она суммирует расстояния каждой плитки от ее целевой позиции. Манхэттенское расстояние — это мера расстояния между двумя точками в сетке, где расстояние между двумя точками рассчитывается как сумма абсолютных различий их координат. В контексте головоломок, манхэттенское расстояние используется для оценки, насколько близко каждая плитка находится к своему целевому положению.

Эвристика по манхэттенскому расстоянию подразумевает, что для каждой плитки подсчитывается манхэттенское расстояние между её текущим положением и её положением в начальном состоянии; полученные величины суммируются. Это даёт оценку общего расстояния от текущего состояния до целевого, что помогает алгоритмам поиска, таким как A\* или IDA\*, определить, какие ходы следует исследовать в первую очередь, чтобы приблизиться к решению головоломки.

Алгоритм IDA\* (Iterative Deepening A\*) — это метод поиска пути в графе, который может определить кратчайший путь между заданным начальным узлом и любым из узлов цели в взвешенном графе.

Функция IDA реализует алгоритм IDA\* для поиска решения. Он использует поиск в глубину с ограничением по глубине, используя эвристику для направления поиска. Алгоритм исследует ходы во всех четырех направлениях (вверх, вниз, влево, вправо) и обновляет состояние доски соответственно. Он отслеживает максимальную достигнутую глубину (mxdep) и путь решения.

Функция main читает конфигурации головоломки из входного файла, проверяет их решаемость и затем пытается решить их с помощью алгоритма IDA\*. Она выводит путь решения, если решение найдено, или сообщение о том, что головоломка не решаема.

**ТЕСТОВЫЕ ДАННЫЕ И РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

На рисунке 3 показаны входные данные, находящиеся в файле.

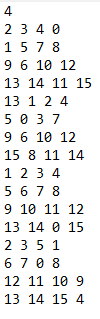


Рисунок 3 — Входные данные

Тестовые данные из методических указаний были пополнены двумя дополнительными тестами, которые можно довольно просто решить самостоятельно и убедиться в правильности работы программы.

Результат обработки указанных входных данных показан на рисунке 4.

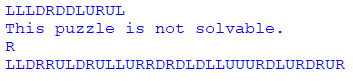


Рисунок 4 — Результат работы программы

Результаты обработки тестов, взятых из методических указаний, сходятся с выводом, указанном там же, что свидетельствует о правильности работы программы. Тесты, добавленные во входной файл самостоятельно, также были обработаны программой верно, что можно проверить простым решением вручную.

**ОПИСАНИЕ ЗАДАЧИ “ВЕСА И МЕРЫ”**

Черепашка по имени Мак очень боится, что его панцирь может сломаться. Поэтому он попросил вас дать совет, как поставить черепашек друг на друга, чтобы построить трон Йертла Черепахи. Каждая из 5607 черепах, призванных Йертлом, обладает своей силой и весом. Ваша задача состоит в том, чтобы составить из них башню максимальной высоты.

**Входные данные**

Входные данные состоят из нескольких строк, каждая из которых содержит два целых числа, разделенные одним или более пробелом. Эти числа задают вес и силу черепахи. Вес черепахи измеряется в граммах. Сила, которая также измеряется в граммах - это максимальный вес, который способна выдержать черепаха (включая свой собственный). Таким образом, черепаха, весящая 300 грамм и имеющая силу 1000 грамм, может держать 700 грамм на спине. Число черепах не превышает 5607.

**Выходные данные**

Выходные данные должны состоять из одного целого числа, равного максимальному числу черепах, которые можно поставить стопкой так, чтобы сила ни одной из них не была превышена.

**Пример входных данных**

300 1000

1000 1200

200 600

100 101

**Соответствующие выходные данные**

3

**АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ “ВЕСА И МЕРЫ”**

Схема алгоритма решения поставленной задачи показана на рисунке 5.

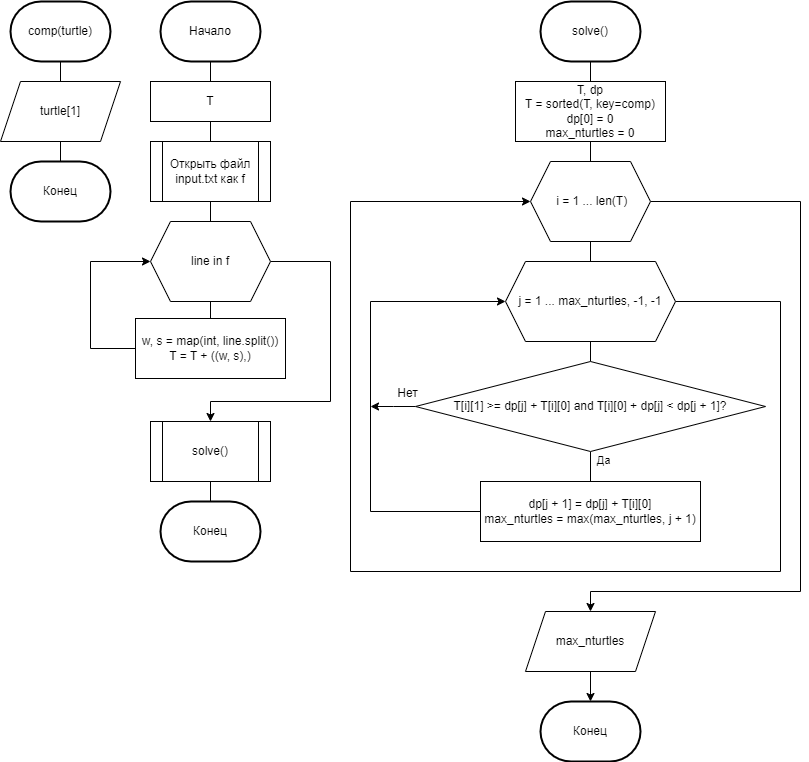


Рисунок 5 — Схема алгоритма решения задачи

Исходный код программы показан на рисунке 6.

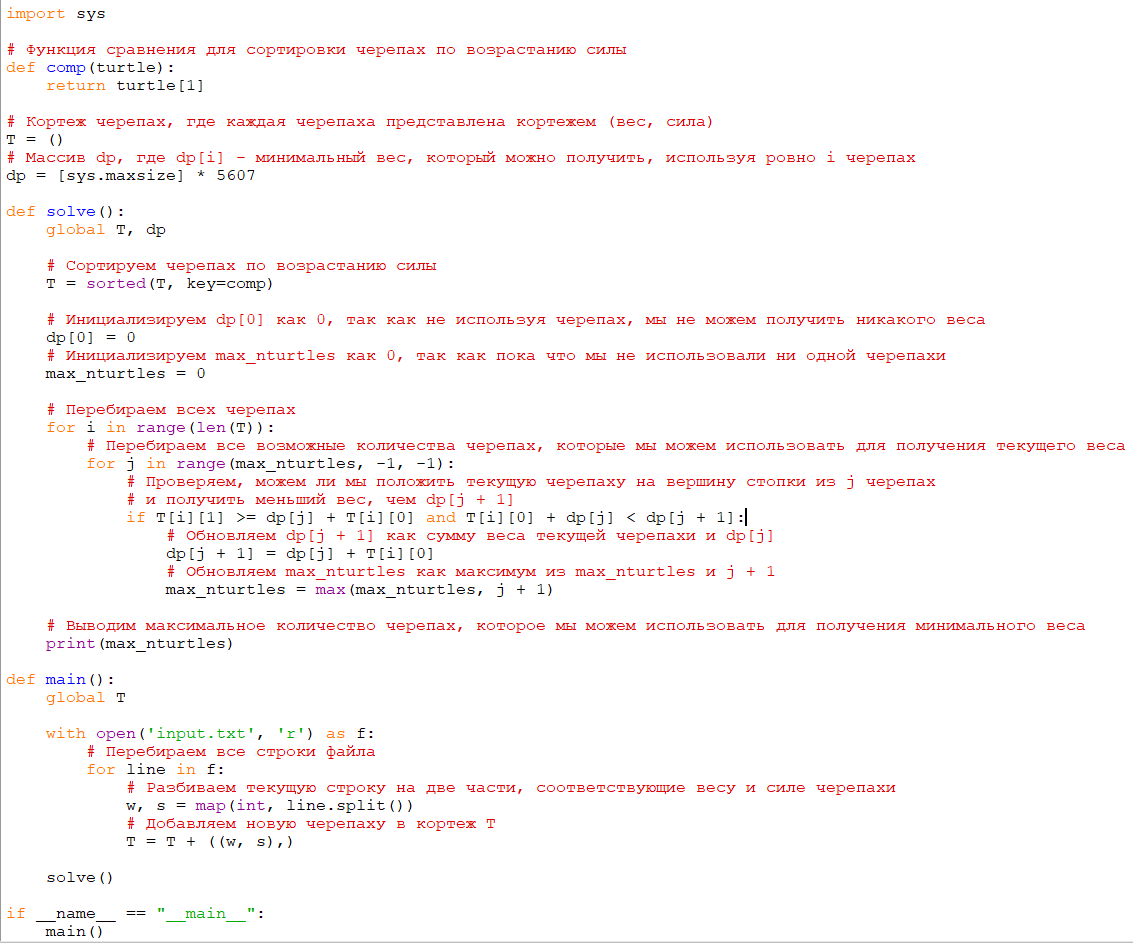


Рисунок 6 — Исходный код решения задачи

Функция comp() используется для сортировки черепах по возрастанию их силы.

Также в коде определены некоторые глобальные переменные:

- Т — кортеж, содержащий кортежи с весом и силой каждой черепахи;

- dp — массив, где dp[i] хранит минимальный вес, который можно получить, используя ровно i черепах;

Функция solve() сортирует черепах по возрастанию силы, инициализирует dp[0] как ноль, так как без использования черепах нельзя получить никакого веса, инициализирует max\_nturtles как 0, т. к. изначально ни одна черепаха не используется, перебирает всех черепах и для каждой черепахи перебирает все возможные количества черепах, которые можно использовать для получения текущего веса, проверяет, можно ли положить текущую черепаху на вершину стопки из j черепах и получить меньший вес, чем dp[j + 1], если условие выполняется, обновляет dp[j + 1] как сумму веса текущей черепахи и dp[j], и обновляет max\_nturtles как максимум из max\_nturtles и j + 1, выводит максимальное количество черепах, которое можно использовать для получения минимального веса.

В функции main() считывает входной файл, добавляет всех черепах в кортеж Т.

**ТЕСТОВЫЕ ДАННЫЕ И РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

На рисунке 7 показаны входные данные, находящиеся в файле.



Рисунок 7 — Входные данные для задачи

Результат обработки входных данных показаны на рисунке 8.



Рисунок 8 — Результат работы программы

Программа верно обрабатывает входные данные, предоставленные в методических указаниях, что показывает правильность работы программы.