|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Языки программирования Python »  Вариант 6 | | |
|  | | Работу выполнил  студент группы ИТ-7,8-2025 1 курса  Комлева Алиса Александровна  «12» июня 2025 г. |
| Работу проверил  Рубцова М. Б.  «16» июня 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 2](#_Toc192898948)

[Алгоритм решения 4](#_Toc792402246)

[Обоснование выбранных структур и типов данных 6](#_Toc2144919390)

[Тестирование 7](#_Toc1442624374)

[Тесты на работоспособность 8](#_Toc992866750)

[Полный список: 8](#_Toc669988185)

[Действие 1: 8](#_Toc471256027)

[Действия 2-3: 9](#_Toc1281615844)

[Выход из программы (действие 4): 10](#_Toc745205389)

[Тесты на проверку “защиты от некорректного пользовательского ввода” 10](#_Toc978767114)

[Код программы 12](#_Toc1810785696)

# Постановка задачи

Для заданного подмножества набора костей домино определить, можно ли их выложить в ряд, не нарушая правил. Если можно, то представить один любой вариант такого разложения. Например, для входных данных 31, 00, 13, получаем ответ: некорректные входные данные; для входных данных 02, 04, 42 ответ: можно, 04, 42, 20. Использовать двухсвязный список.

# Алгоритм решения

1. Проверка входных данных (класс DominoSolver, методы is\_valid\_domino и validate\_input).

Алгоритм начинает работу с тщательной проверки входных данных через два ключевых метода:

Метод is\_valid\_domino(domino):

• Проверяет длину строки: len(domino) == 2

• Проверяет, что оба символа - цифры: domino[0].isdigit() and domino[1].isdigit()

• Проверяет диапазон значений: 0 <= int(цифра) <= 6 для обеих цифр

• Возвращает False при любом нарушении этих условий

Метод validate\_input():

• Проверяет, что список костей не пуст

• Для каждой кости вызывает is\_valid\_domino()

• При обнаружении некорректных данных сразу возвращает False

1. Поиск цепочки (класс DominoSolver, методы can\_connect и find\_chain).

Основная логика поиска реализована с применением рекурсивного backtracking:

Метод can\_connect(a, b, reverse=False):

• Проверяет возможность соединения двух костей

• При reverse=False: проверяет a[1] == b[0]

• При reverse=True: проверяет a[1] == b[1] (кость b перевернута)

Метод find\_chain(current\_list):

• Базовый случай: если все кости использованы (all(self.used)), сохраняет решение

• Рекурсивный случай:

• Берет последнюю кость из current\_list

• Перебирает все неиспользованные кости

• Для каждой проверяет возможность соединения в обычном и перевернутом виде

• При успешном соединении - рекурсивный вызов find\_chain

• При неудаче - откат изменений (backtrack)

1. Вывод результата (класс DominoSolver, метод solve).

Метод solve() управляет всем процессом:

• Проверяет входные данные через validate\_input()

• Если данные некорректны - сразу возвращает ошибку

• Для каждой кости пытается построить цепочку:

• В обычном и перевернутом виде

• При успехе возвращает найденную цепочку

• Если ни одна попытка не удалась - возвращает "нельзя выложить в ряд"

1. Работа с пользователем (функции print\_menu, show\_examples, about\_program, main).

Интерактивное меню реализовано через несколько функций:

print\_menu():

• Выводит красиво оформленное меню с 4 пунктами

• Обрамлено символами "=" для визуального выделения

show\_examples():

• Показывает примеры корректного и некорректного ввода

• Демонстрирует возможные результаты работы программы

about\_program():

• Выводит краткую информацию о назначении программы

• Описывает основные правила работы с домино

main():

• Основной цикл программы

• Обрабатывает выбор пользователя

• Вызывает соответствующие функции

• Обеспечивает обработку ошибок ввода

## Обоснование выбранных структур и типов данных

1. DominoNode (узел двусвязного списка):

Хранит:

value (кость, строка из двух цифр),

prev (ссылка на предыдущий узел),

next (ссылка на следующий узел).

Обоснование: Удобен для динамического построения цепочки и backtracking.

1. DominoList (двусвязный список):

Поддерживает операции:

append (добавление в конец),

to\_list (преобразование в стандартный список),

copy (глубокое копирование).

Обоснование:Позволяет эффективно добавлять и удалять элементы при переборе вариантов.

1. DominoSolver (основной алгоритм):

Использует:

Массив used (отметки об использованных костях),

Рекурсивный backtracking для поиска цепочки.

Обоснование:

used предотвращает повторное использование костей.

Backtracking гарантирует полный перебор всех возможных комбинаций.

# Тестирование

## Тесты на работоспособность

## Полный список:

Добро пожаловать в программу проверки домино!

Цифры на костях должны быть от 0 до 6 (включительно).

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4):

## Действие 1:

Добро пожаловать в программу проверки домино!

Цифры на костях должны быть от 0 до 6 (включительно).

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 1

Введите кости домино через запятую: 02,04,24

Результат: можно, 02, 24, 40

Нажмите Enter чтобы продолжить...

## Действия 2-3:

Добро пожаловать в программу проверки домино!

Цифры на костях должны быть от 0 до 6 (включительно).

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 2

Примеры корректного ввода:

- 02, 04, 42 → можно, 04, 42, 20

- 11, 22, 33 → нельзя выложить в ряд

Примеры некорректного ввода:

- 31, 00, 13 → некорректные входные данные

- 7, 12, 34 → цифры должны быть от 0 до 6

- abc, 12 → должно быть две цифры

Нажмите Enter чтобы продолжить...

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 3

Программа для проверки возможности выстроить

ряд из заданных костей домино по правилам:

- соседние кости должны соприкасаться одинаковыми числами

- кости можно переворачивать

Нажмите Enter чтобы продолжить...

## Выход из программы (действие 4):

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 4

Спасибо за использование программы. До свидания!

## Тесты на проверку “защиты от некорректного пользовательского ввода”

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 5

Ошибка: выберите пункт меню от 1 до 4

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): муик

Ошибка: выберите пункт меню от 1 до 4

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 1

Введите кости домино через запятую: 01 02 24

Результат: некорректные входные данные

Нажмите Enter чтобы продолжить...

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 1

Введите кости домино через запятую: 01,02,77

Результат: некорректные входные данные

Нажмите Enter чтобы продолжить...

==================================================

МЕНЮ ПРОГРАММЫ ДОМИНО

==================================================

1. Проверить возможность построения цепочки

2. Показать примеры ввода

3. О программе

4. Выход

==================================================

Выберите пункт меню (1-4): 1

Введите кости домино через запятую: 111,02,01

Результат: некорректные входные данные

Нажмите Enter чтобы продолжить...

# Код программы

[Код](https://github.com/Komlevaalisa/-.git)