Министерство высшего образования и науки РФ

Санкт–Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчет**

по дисциплине «Системный подход в разработке ПО»

**Коллекции**

**Выполнил:**

студент группы 5130902/20201 \_\_\_\_\_\_\_\_\_\_\_\_ \_\_ А. И. Сафонов

подпись

**Проверил:**

доцент, к.т.н \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Нестеров

подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г.

Санкт-Петербург

2025

Оглавление

[1. Задания по 4.1 3](#_Toc193051574)

[1.1. Вывод строки в консоль 3](#_Toc193051575)

[1.2. Вывод задом наперед 4](#_Toc193051576)

[1.3. Поиск символа 5](#_Toc193051577)

[1.4. Парсинг строк в список 6](#_Toc193051578)

[1.5. Разбитие строки 7](#_Toc193051579)

[1.6. Сравнение String, StringBuilder 8](#_Toc193051580)

[1.7. Пример StringBuilder 9](#_Toc193051581)

[1.8. Больше методов с StringBuilder 10](#_Toc193051582)

[1.9. indexOf 11](#_Toc193051583)

[2. Практика 4.1 12](#_Toc193051584)

[3. Задания 4.2 16](#_Toc193051585)

[3.1. Regexp 16](#_Toc193051586)

[3.2. Рифма 17](#_Toc193051587)

[3.3. Валидация 19](#_Toc193051588)

# Задания по 4.1

## Вывод строки в консоль

A screen shot of a computer program

AI-generated content may be incorrect.

## Вывод задом наперед

A screen shot of a computer

AI-generated content may be incorrect.

## Поиск символа

A screen shot of a computer

AI-generated content may be incorrect.

## Парсинг строк в список

A screen shot of a computer program

AI-generated content may be incorrect.

## Разбитие строки

A screen shot of a computer program

AI-generated content may be incorrect.

## Сравнение String, StringBuilder

A screen shot of a computer

AI-generated content may be incorrect.

## Пример StringBuilder

A screen shot of a computer program

AI-generated content may be incorrect.

## Больше методов с StringBuilder

A screenshot of a computer program

AI-generated content may be incorrect.

## indexOf

A screenshot of a computer program

AI-generated content may be incorrect.

# Практика 4.1

Изображение выглядит как текст, линия, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

1. Splitting
2. StringBuilder
3. Substring

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

What is the major difference between making modifications to a String and a StringBuilder?

String creates new object while StringBuilder changes one.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

# Задания 4.2

## Regexp

A screen shot of a computer program

AI-generated content may be incorrect.

## Рифма

A screen shot of a computer program

AI-generated content may be incorrect.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

## Валидация

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

# Васильев

Итак, что происходит, если при выполнении программы возникает ошибка? Во-первых, выполнение программы приостанавливается. Во-вторых, Глава 11 398 создается объект, который содержит информацию об ошибке, которая возникла. Поскольку объекты создаются на основе классов, то возникает естественный вопрос: на основе какого класса создается объект ошибки? Ответ состоит в том, что в Java есть иерархия классов, каждый из которых соответствует ошибке определенного типа. В вершине этой иерархии находится класс Throwable . У класса Throwable имеется два подкласса: Error и Exception. Класс Error соответствует «фатальным» ошибкам, которые обычно программными методами не обрабатываются. Нас интересует класс Exception. У него очень много подклассов, и среди этих подклассов есть класс RuntimeException . В свою очередь, класс RuntimeException является суперклассом для классов, описывающих всевозможные ошибки, возникающие в процессе выполнения программного кода (классы обсуждаются далее). Собственно это та часть иерархии классов исключений (ошибок), которая для нас представляет наибольший интерес. Итак, если ошибка возникла, то в соответствии с ее типом создается объект. Этот объект (исключение) передается для обработки методу, который вызвал ошибку (говорят, что исключение вбрасывается в метод). Если в методе ошибка обрабатывается — хорошо. Если ошибка в методе не обрабатывается, то объект ошибки передается для обработки в метод, в котором вызывался данный метод (вызвавший ошибку), и так по цепочке, вплоть до главного метода программы. Если и в главном методе ошибка не обрабатывается, то в дело вступает так называемый обработчик по умолчанию. Неприятное последствие задействования обработчика по умолчанию в том, что программа досрочно прекращает выполнение. Теперь рассмотрим процесс обработки исключений. Если мы «подозреваем» некоторый фрагмент кода на предмет того, что он может вызвать ошибку, то такой фрагмент кода помещается в try-блок. После try-блока следуют catch-блоки (и может быть несколько). Каждый catch-блок предназначен для обработки ошибок определенного типа. Под типом ошибки в данном случае имеется в виду класс, на основе которого, в случае возникновения ошибки, создается объект ошибки. Некоторые наиболее актуальные с практической точки зрения классы ошибок перечислены в табл.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

В каждом catch-блоке в круглых скобках кроме класса ошибки указывается формальное название для объекта ошибки. При необходимости, данный объект может быть использован в явном виде в программном коде catch-блока. Если при выполнении программного кода в try-блоке ошибка не возникает, то все catch-блоки игнорируются. Если в try-блоке возникла ошибка, то, как отмечалось ранее, для данной ошибки создается объект, и далее с целью обработки исключения проверяются catch-блоки. Проверка выполняется на предмет совпадения класса объекта ошибки и класса, указанного в catch-блоке. Если совпадение найдено, объект ошибки передается в соответствующий catch-блок для обработки. Важное обстоятельство состоит в том, что «совпадение» понимается в широком смысле: классы могут совпадать не буквально — достаточно, чтобы класс объекта ошибки был подклассом класса, указанного в catch-блоке. В этом смысле, если мы в catch-блоке указываем имя класса Exception, то такой catch-блок будет перехватывать фактически все ошибки (которые в принципе можно обработать), поскольку классы ошибок, с которыми мы имеем дело, прямо или опосредованно являются наследниками класса Exception.

Блоки try-catch могут быть вложенными. В таком случае если ошибка возникает во внутреннем try-блоке, то для обработки ошибки сначала просматриваются catch-блоки внутреннего try-блока (в котором возникла ошибка). Если ошибка внутренними catch-блоками не обрабатывается, то исключение передается для обработки во внешних catch-блоках.

В описании catch-блоков неизменно присутствует, кроме имени класса обрабатываемого в блоке исключения, еще и формальное обозначение для объекта исключения . Эти объекты можно использовать при обработке ошибок. В частности, объект исключения можно использовать в выражениях в качестве «текстового» операнда: благодаря переопределенному методу toString() объект исключения в таких случаях автоматически приводится к текстовому формату. Полученное текстовое значение содержит краткую, но все же полезную информацию о возникшей ошибке

Хотя на первый взгляд это и выглядит несколько странным, но исключения можно генерировать , так сказать, «вручную». В частности, для искусственного генерирования исключений используется оператор throw , после которого указывается объект исключения. Объект исключения создают специально (как создается любой другой объект), или использую уже сгенерированный объект, переданный для обработки в catchблок

Все исключения в Java делятся на контролируемые и неконтролируемые. К неконтролируемым исключениям относятся исключения классов, являющихся подклассами класса RuntimeException или класса Error . Все остальные исключения относятся к контролируемым. Разница между исключениями разных типов состоит в том, что для контролируемых исключений автоматически выполняется проверка на наличие обработки исключения. Другими словами, если метод может сгенерировать в процессе выполнения контролируемое исключение, то для такого исключения в методе должна быть предусмотрена обработка. Если метод все же не содержит код для обработки контролируемого исключения (но потенциально может такое исключение сгенерировать), то в описании метода в явном виде указывается, что он может Обработка исключительных ситуаций 417 генерировать необрабатываемое контролируемое исключение. Делается это просто: в описании метода после его имени и списка аргументов (но перед фигурными скобками с кодом метода) указывается ключевое слово throws и через запятую перечисляются классы контролируемых исключений, которые метод может сгенерировать, но которые в теле метода не обрабатываются

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Существует возможность создавать пользовательские классы для исключений . Рецепт простой: на основе одного из классов исключений путем наследования создается класс. Объект, созданный на основе такого класса, может использоваться при генерировании исключительных ситуаций. Например, в качестве суперкласса можем использовать класс Exception , или, скажем, класс RuntimeException . В первом случае получим пользовательский класс исключения контролируемого типа, а во втором — неконтролируемого типа.

## Пример try catch

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Контент, сгенерированный ИИ, может содержать ошибки.

## Размер массива

Изображение выглядит как текст, программное обеспечение, снимок экрана, компьютер

Контент, сгенерированный ИИ, может содержать ошибки.

При отрицательном значении

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

При вводе строки

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

## Вложенный с массивом

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Объекты исключений

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Проброс исключения

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Проверяемые исключения

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

## Кастомные исключения

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Конспект главы «Связные списки»**

**1. Основные понятия**

**Ячейка и ссылки.** Связный список состоит из ячеек, каждая из которых содержит данные и ссылку (указатель) на следующую ячейку. В примере на C# создаётся класс, где поле Next указывает на следующую ячейку.

* **Ограничитель (сентинел).** Часто в начале (и/или в конце) списка используется специальная ячейка – ограничитель, которая упрощает алгоритмы (например, позволяет избежать проверки на null в начале).

**2. Однонаправленные списки**

* **Перебор и поиск.** Для обхода списка используется цикл, который последовательно переходит по ссылкам (например, алгоритм Iterate, где каждый элемент обрабатывается, пока ссылка не станет null). Поиск элемента проходит путём последовательного сравнения значений.
* **Добавление и вставка.**
  + **Добавление в начало:** Новый элемент вставляется сразу после ограничителя за время O(1).
  + **Добавление в конец:** Сначала требуется пройти весь список (O(N)), затем добавить элемент.
  + **Вставка в середину:** Находится нужная позиция (при необходимости — сначала выполняется поиск), после чего выполняется стандартная операция вставки.

**3. Двунаправленные списки**

* Каждая ячейка содержит две ссылки: на следующий и на предыдущий элементы.
* Благодаря наличию ссылок в обоих направлениях операции вставки и удаления можно выполнять за O(1) при наличии нужных указателей, а ограничители (верхний и нижний) упрощают работу с концами списка.

**4. Дополнительные операции**

* **Копирование списка.** Для создания копии списка создаётся новый ограничитель, а затем последовательно копируются все ячейки.
* **Сортировка.** Рассматриваются два метода:
  + **Сортировка вставкой.** Из неотсортированного списка элементы последовательно вставляются в новый список в нужном порядке.
  + **Сортировка выбором.** На каждом шаге из оставшихся элементов выбирается наибольший и перемещается в начало нового списка.
* **Многопотоковые списки.** Один набор ячеек может поддерживать несколько «потоков» (например, планеты можно сортировать по удалённости от Солнца, массе, диаметру и т. п.) с использованием дополнительных ссылок.

**5. Работа с циклами в списках**

* **Кольцевой список.** Последняя ячейка может указывать на первый элемент, что создаёт цикл. Такие структуры полезны для повторяющихся циклических операций.
* **Обнаружение циклов.** Рассмотрены несколько методов:
  + **Маркировка ячеек.** Каждая ячейка получает флаг (Visited), если флаг уже установлен — найден цикл.
  + **Хеш-таблица.** При обходе списка сохраняются посещённые ячейки в хеш-таблице; обнаружение повторного вхождения указывает на цикл.
  + **Повторная трассировка.** Для каждой ячейки повторно проверяется, не ссылается ли предыдущая ячейка на ту же ячейку, что и текущая.
  + **Реверсирование списка.** Смена направлений ссылок позволяет по восстановлению первоначального порядка определить наличие цикла.
  + **Алгоритм черепахи и кролика (Флойд).** Два указателя движутся с разной скоростью: если список цикличен, они встретятся внутри цикла. После встречи один из указателей перезапускается с начала, и повторная встреча указывает на начало цикла.

**Упражнение 1.**  
*Задача:* Добавить переменную bottom, указывающую на последнюю ячейку, для ускорения добавления в конец (O(1)).  
*Разбор:*

* При использовании bottom операция вставки в конец становится постоянной по времени, однако нужно поддерживать корректность этой переменной при операциях вставки/удаления с начала и в середине списка.
* При удалении элемента из конца потребуется обновлять указатель bottom — если удаляется последний элемент, нужно пройтись по списку, чтобы найти новый последний элемент, или хранить указатель на предыдущий элемент в каждой ячейке.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 2.**  
*Задача:* Найти наибольший элемент в несортированном однонаправленном списке (ячейки содержат целые числа).  
*Разбор:*

* Алгоритм простой: инициализировать переменную с первым значением, затем пройти список, сравнивая текущее максимальное значение с каждым элементом.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 3.**  
*Задача:* Написать алгоритм для добавления элемента в начало двунаправленного списка.  
*Разбор:*

* Вставка в начало аналогична однонаправленному списку, но требуется обновить ссылки Prev и Next. Новый элемент должен стать первым, а его Next указывать на бывший первый элемент, у которого Prev надо обновить на новый элемент.

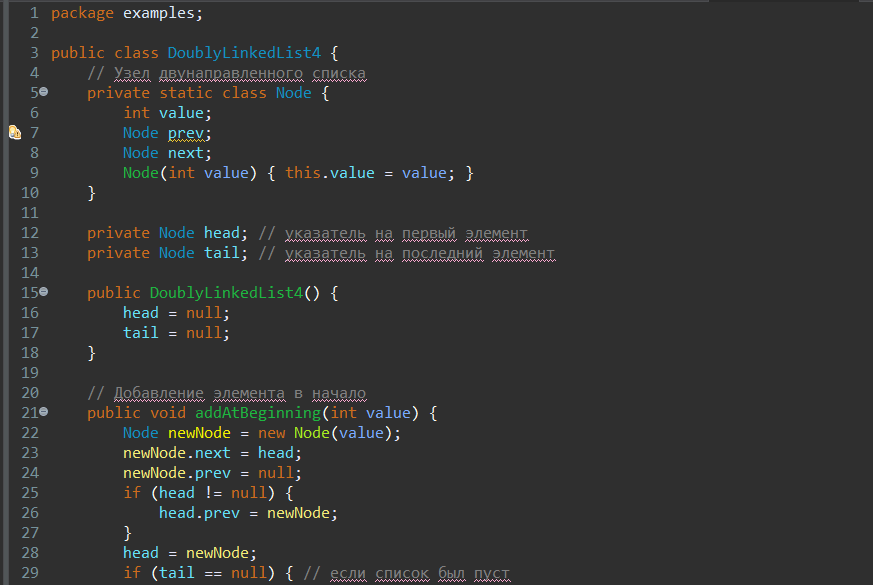
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 4.**  
*Задача:* Создать алгоритм для добавления элемента в конец двунаправленного списка.  
*Разбор:*

* Если есть нижний ограничитель (или указатель на конец), то вставка производится аналогично вставке в конец однонаправленного списка, с добавлением обновления обратных ссылок (Prev).

 Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 5.**  
*Задача:* Переписать алгоритмы из упражнений 3 и 4 так, чтобы они вызывали уже написанный алгоритм InsertCell для двунаправленных списков, вместо прямого обновления ссылок.  
*Разбор:*

* Нужно вынести логику обновления ссылок (как в примере с InsertCell) в отдельную функцию и вызывать её из функций для добавления в начало и конец. Это позволит избежать дублирования кода.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 6.**  
*Задача:* Написать алгоритм удаления конкретной ячейки из двунаправленного списка и представить процесс в виде рисунка.  
*Разбор:*

* Для удаления ячейки требуется обновить Next у предыдущей ячейки и Prev у следующей. Рисунок может показывать три ячейки: предыдущую, удаляемую и следующую, с обновлением стрелок после удаления.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 7.**  
*Задача:* В отсортированном двунаправленном списке с именами можно ли ускорить поиск, начиная не с верхнего, а с нижнего ограничителя?  
*Разбор:*

* Если список отсортирован (например, по алфавиту), то выбор стороны обхода (начало или конец) может уменьшить количество шагов, если искомый элемент ближе к соответствующему концу. Однако асимптотическая сложность остаётся O(N), но на практике может наблюдаться ускорение.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 8.**  
*Задача:* Написать алгоритм вставки элемента в сортированный двунаправленный список с использованием верхнего и нижнего ограничителей, содержащих минимальное и максимальное значения.  
*Разбор:*

* Ограничители гарантируют, что вставка всегда найдёт нужное место без дополнительной проверки на конец списка. Алгоритм сравнивает значение нового элемента с элементами списка, пока не найдёт правильное место, и затем вызывает универсальную функцию вставки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 9.**  
*Задача:* Реализовать алгоритм, определяющий, отсортирован ли связный список.  
*Разбор:*

* Пройти по списку, сравнивая значение текущей ячейки со следующим. Если найдётся пара, где порядок нарушен (например, текущий больше следующего для сортировки по возрастанию), то список не отсортирован.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 10.**  
*Задача:* Объяснить, почему сортировка выбором на практике работает медленнее сортировки вставкой, несмотря на одинаковую асимптотику O(N²).  
*Разбор:*

* При сортировке выбором за каждый проход требуется находить наибольший (или наименьший) элемент, что предполагает больше сравнений и, возможно, больше операций копирования/перестановки. Сортировка вставкой может быстрее работать на почти отсортированных данных, что часто встречается на практике.

Обе сортировки имеют квадратичную сложность в худшем случае, но на практике insertion sort может выполнять меньше перестановок, особенно если список почти отсортирован.

В сортировке выбором на каждом шаге необходимо просматривать весь неотсортированный фрагмент для нахождения минимального (или максимального) элемента, что приводит к большему числу сравнений.

Кроме того, в выборочной сортировке перестановки выполняются независимо от начального порядка, тогда как сортировка вставкой "подстраивается" под уже частично отсортированный список.

Таким образом, на реальных данных алгоритм сортировки вставкой зачастую оказывается быстрее.

**Упражнение 11.**  
*Задача:* Создать программу, которая формирует многопотоковый список планет.  
*Разбор:*

* Необходимо создать класс Planet с полями для хранения имени, расстояния до Солнца, массы, диаметра и дополнительных указателей для каждого «потока» сортировки. Метод AddPlanetToList должен вставлять планету в каждый из потоков с учётом сортировки (например, по расстоянию, массе и диаметру).
* Здесь важно продумать архитектуру, чтобы пользователь мог выбирать нужный критерий сортировки через, например, выпадающий список.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Упражнение 12.**  
*Задача:* Написать программу, реализующую алгоритм черепахи и кролика для обнаружения цикла.  
*Разбор:*

* Реализация включает запуск двух указателей – один перемещается на одну ячейку за шаг (черепаха), другой – на две (кролик). Если список цикличен, указатели встретятся в цикле. После встречи один из указателей перемещается с начала списка, и новая встреча укажет на начало цикла, что позволит разорвать цикл.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, Шрифт, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.