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 «Язык программирования Python»
Вариант 12

Работу выполнила
Студентка группы ИТ-8 ,2025 1 курса
Рязанова Кристина Михайловна.
«12» июня 2025 г.

Работу проверила
Рубцова Марина Борисовна
«___» _____ 2025 г.

Пермь 2025

СОДЕРЖАНИЕ

Постановка задачи 3

Алгоритм решения..... 4-5

Тестирование..... 6-7

Код программы 7

Инструкция по применению стилей и оформлению работы.....8

Постановка задачи

1.Шарики. В одной компьютерной игре игрок выставляет в линию шарики разных цветов. Когда образуется непрерывная цепочка из трех и более шариков одного цвета, она удаляется из линии. Все шарики при этом сдвигаются друг к другу, и ситуация может повториться. Напишите программу, которая по данной ситуации определяет, сколько шариков будет сейчас уничтожено. Естественно, непрерывных цепочек из трех и более одноцветных шаров в начальный момент может быть не более одной. Входные данные: даны количество шариков в цепочке (не более 105) и цвета шариков (от 0 до 9, каждому цвету соответствует свое целое число). Выходные данные: требуется вывести количество шариков, которое будет уничтожено.
Примеры:
(входные данные) 5 1 3 3 3 2 (выходные данные) 3
(входные данные) 10 3 3 2 1 1 1 2 2 3 3 (выходные данные) 10

2.Определение алгоритма и идеи решения поставленной задачи
Все задачи подразумевают работу с динамическими структурами данных – списками (односвязные/двусвязные/циклические/стеки/очереди …) или деревьями. Для реализации используем собственные классы. Специализированные библиотеки использовать нельзя! Обязательно наличие обработки исключений.

3.Написание и структурирование кода решения задачи
При написании кода следует придерживаться методологии ООП. Несоответствие код стайлу , принятому в группе, может быть основанием для оценивании работы в 0 баллов. В программе обязательно наличие дружественного интерфейса и защиты от некорректного пользовательского ввода.

Алгоритм решения

1. Инициализация:
- Класс Node предназначен для создания узла цепочки, имеющего цвет и ссылку на следующий узел.
- Класс BallChain инициализирует цепочку на основе входных данных, создавая узлы Node для каждого цвета шарика.

2. Создание цепочки:
- Метод build_chain проходит по списку цветов и создает узлы. Если цепочка пустая, новый узел становится её головой; если нет — метод переходит к последнему узлу и добавляет новый узел в конец.

3.Обработка цепочки:
- Метод process_chain запускает цикл, удаляющий последовательности одинаковых шариков, пока это возможно. Он вызывает remove_sequences для удаления последовательностей и подсчитывает общее количество удаленных шариков.

4. Удаление последовательностей:
- Метод remove_sequences ищет последовательности из 3 и более одинаковых шариков. Если такие находятся, они удаляются из цепочки. Метод обновляет ссылки предыдущего узла и возвращает количество удаленных шариков.

5.Пользовательский интерфейс:
- В функции main реализован интерфейс для ввода данных пользователем. Она обрабатывает ввод, проверяет валидность данных, создает цепочку и выводит количество удаленных шариков, а затем предлагает пользователю ввести новые данные.

Обоснование выбранных структур и типов данных :

1. Класс Node:
- Структура Node используется для представления узлов в виде связного списка, что обеспечивает динамическое выделение памяти и удобную реализацию операций вставки и удаления.

2. Списки:
- Входные данные (цвета шариков) хранятся в списке, что позволяет удобно работать с последовательностями, проходить по ним и применять различные условия.

3.Циклы и условия:

- Используемые циклы и условия в методах build_chain и remove_sequences позволяют эффективно обрабатывать данные, обеспечивая минимальное время выполнения операций по сравнению с другими структурами данных.

4. Обработка ошибок:
- В функции main предусмотрена обработка ошибок для обеспечения надежности — пользователь может получить инструкции при неверном вводе.

Обработка исключений

1. Проверка корректности пользовательского ввода: Код ожидает, что пользователь введет числа, разделенные пробелами. Однако, может произойти ошибка, если пользователь введет неверные данные. Блок try-except позволяет программе перехватить такое исключение (ValueError) и продолжить выполнение, предлагая пользователю корректный ввод.

2. На уровне логики программы: Обработка исключений помогает структурировать поведение программы. В случае неправильного ввода сразу же выдается сообщение об ошибке, что делает интерфейс более дружелюбным.

3. Избежание неожиданного завершения: Вместо того чтобы программа завершалась с ошибкой, она продолжает работать, позволяя пользователю повторить попытку ввода.

Классы

1. Класс Node:
- Инкапсуляция: Этот класс представляет собой элемент цепочки (узел), содержащий цвета шариков и ссылку на следующий узел. Данные (цвет и ссылка на следующий узел) инкапсулированы внутри класса, обеспечивая защиту от некорректного доступа.

2. Класс BallChain:
- Инкапсуляция: Данные о цепочке (в данном случае head) находятся внутри класса. Методы, такие как build_chain, process_chain и remove_sequences, инкапсулируют логику работы с этими данными. Это облегчает поддержку кода и делает его более читабельным.

Отсутствие наследования и полиморфизма - данный код не демонстрирует явного полиморфизма, поскольку нет наследуемых классов

Использование классов и методов позволяет разбивать код на логические части, что улучшает его читаемость и поддержку

Тестирование

Тесты с корректным вводом:

```
#1.тест на удаление одной цепочки
#Программа подсчета удаляемых шариков
#Введите количество шариков и их цвета через пробел
#Пример: 5 1 3 3 3 2
#Введите данные: 5 1 3 3 3 2
#Результат: 3
#Хотите проверить другую цепочку? (да/нет): да
#Введите данные:
#2.тест на удаление нескольких цепочек
#Введите данные: 10 3 3 2 1 1 1 2 2 3 3
#Результат: 10
#3.тест на отсутствие цепочек
#Введите данные: 5 1 2 3 4 5
#Результат: 0
#4.тест на короткую цепочку(менее 3 шариков)
#Введите данные: 4 1 1 2 3
#Результат: 0
#5.тест на все шарики одного цвета
#Введите данные: 5 1 1 1 1 1
#Результат: 5
#6.тест на цепочку в конце списка
#Введите данные: 6 1 2 3 4 4 4
#Результат: 3
#7.тест на цепочку в начале списка
#Введите данные: 6 5 5 5 1 2 3
#Результат: 3
```

Тесты с некорректным вводом:

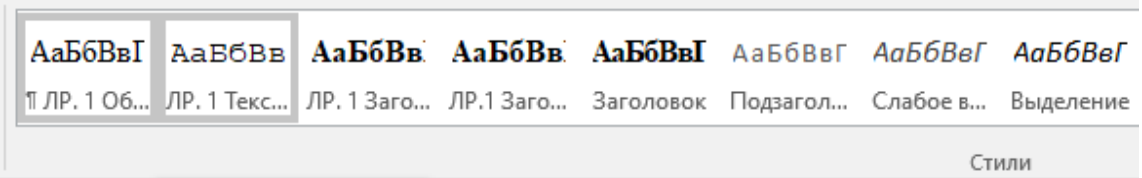
```
#8.тест на пустой ввод
#Введите данные:
#Ошибка: ввод не может быть пустым. Попробуйте снова.
#Введите данные:
#9.тест на неправильное количество шариков
#Введите данные: 4 1 2 2 3 3
#Ошибка: указано 4 шариков, но введено 5 цветов. Попробуйте снова.
#10.тест на ввод не целых чисел
#Введите данные: 4 a b c d
#Ошибка: все вводимые значения должны быть целыми числами. Попробуйте снова.
#11.тест на ввод шариков не от 0 до 9
#Введите данные: 3 11 45 22
#Ошибка: цвета шариков должны быть числами от 0 до 9. Попробуйте снова.
#12.тест
#Введите данные: 0
#Ошибка: нужно ввести как минимум количество шариков и один цвет. Попробуйте снова.
```

Код программы

https://github.com/Kriistarz/ikm

Инструкция по применению стилей и оформлению работы

Для оформления частей отчёта следует использовать заранее созданные стили. Все стили, которые могут пригодиться, начинаются с «ЛР. 1 …».



ЛР. 1 Обычный – для оформления текста задания и алгоритма решения.

ЛР. 1 Текст программы – для оформления кода программы.

ЛР. 1 Заголовок 1 – заголовок первого уровня (для того, чтобы озаглавить основные разделы отчета).

ЛР. 1 Заголовок 2 – заголовок второго уровня (для того, чтобы озаглавить подразделы).

Для того, чтобы перенести текст следующего блока на другую страницу, необходимо воспользоваться инструментов «Разрыв страницы» в разделе «Вставка».