

УДК 004.81

<https://doi.org/10.21869/2223-1560-2025-29-2-146-165>

Автоматизированная генерация презентаций на основе когнитивно-эстетической парадигмы

А. А. Зоткина ¹ ✉, А. И. Мартышкин ¹, А. А. Павлов ¹, А. В. Ткаченко ¹

¹ Пензенский государственный технологический университет
пр. Байдукова/ул. Гагарина, д. 1а/11, г. Пенза 440039, Российская Федерация

✉ e-mail: alena.zotkina.97@mail.ru

Резюме

Цель исследования. Разработка программного решения для автоматизированной генерации презентаций, ориентированного на интеграцию принципов когнитивной психологии и эстетики. Исследование направлено на создание интеллектуальной платформы, способной формировать презентации, которые не только передают информацию, но и обеспечивают её эффективное визуальное восприятие, привлекая и удерживая внимание аудитории. В условиях стремительного роста объёмов данных и сокращения временных ресурсов возрастает потребность в инструментах, способных автоматизировать процесс создания визуального контента без ущерба для его качества. Предлагаемое решение учитывает особенности восприятия информации человеком, опираясь на такие когнитивные принципы, как теория когнитивной нагрузки, гештальт-принципы и визуальное выделение, а также применяет методы эстетического проектирования для создания сбалансированной и выразительной визуальной структуры.

Методы. Статья сочетает различные подходы и методы, включая когнитивные и эстетические аспекты, объектно-ориентированное программирование и интеграцию с технологиями искусственного интеллекта, что позволяет автоматически формировать как текстовый, так и визуальный контент, адаптируя его под заданную структуру презентации.

Результаты. Разработанное программное решение продемонстрировало преимущество по сравнению с существующими аналогами в ряде параметров: сокращение времени на создание презентаций, повышение визуального качества, адаптивность и соответствие когнитивным стратегиям восприятия информации. Результаты эксперимента с участием пользователей подтвердили высокую эффективность предложенного подхода по критериям удобства использования, дизайна и восприятия.

Заключение. Предложенное решение, основанное на когнитивно-эстетической парадигме, представляет собой перспективное направление в области автоматизации визуальной коммуникации. Оно демонстрирует потенциал к масштабируемости и интеграции в различные сферы, где требуется создание презентационного контента, совмещающего информативность, визуальную выразительность и человеко-ориентированный подход.

Ключевые слова: генерация; презентация; когнитивно-эстетическая парадигма; Python; LLM и TTI.

Конфликт интересов: Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.

© Зоткина А. А., Мартышкин А. И., Павлов А. А., Ткаченко А. В., 2025

Для цитирования: Автоматизированная генерация презентаций на основе когнитивно-эстетической парадигмы / А. А. Зоткина, А. И. Мартышкин, А. А. Павлов, А. В. Ткаченко // Известия Юго-Западного государственного университета. 2025; 29(2): 146-165. <https://doi.org/10.21869/2223-1560-2025-29-2-146-165>.

Поступила в редакцию 02.04.2025

Подписана в печать 29.05.2025

Опубликована 23.07.2025

Automated generation of presentations based on the cognitive-aesthetic paradigm

Alena A. Zotkina ¹ ✉, Alexey I. Martyshkin ¹, Akim A. Pavlov ¹,
Alexandra V. Tkachenko ¹

¹ Penza State Technological University
1a/11, Baidukova ave. / Gagarina str., Penza 440039, Russian Federation

✉ e-mail: alena.zotkina.97@mail.ru

Abstract

Purpose of research. The objective of this study is to develop a software solution for automated presentation generation, focused on integrating principles of cognitive psychology and aesthetics. The research aims to create an intelligent platform capable of generating presentations that not only convey information but also ensure its effective visual perception, capturing and retaining audience attention. In an era of rapidly increasing data volumes and decreasing time resources, there is a growing need for tools that can automate the creation of visual content without compromising its quality. The proposed solution takes into account the specific characteristics of human information perception by relying on cognitive principles such as cognitive load theory, Gestalt laws, and visual emphasis. In addition, it applies aesthetic design techniques to create a balanced and expressive visual structure.

Methods. The study combines various approaches and methods, including cognitive and aesthetic aspects, object-oriented programming, and integration with artificial intelligence technologies. This allows the system to automatically generate both textual and visual content, adapting it to the predefined structure of the presentation.

Results. The developed software solution demonstrated advantages over existing tools across several key parameters: reduced presentation creation time, improved visual quality, adaptability, and alignment with cognitive strategies of information perception. Experimental results involving users confirmed the high efficiency of the proposed approach in terms of usability, design quality, and content perception.

Conclusion. The proposed solution, based on the cognitive-aesthetic paradigm, represents a promising direction in the field of automated visual communication. It demonstrates potential for scalability and integration across various domains that require the creation of presentation content combining informativeness, visual expressiveness, and a human-centered design approach.

Keywords: generation; presentation; cognitive-aesthetic paradigm; Python; LLM and TTI.

Conflict of interest. The Authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.

For citation: Zotkina A. A., Martyshkin A. I., Pavlov A. A., Tkachenko A. V. Automated generation of presentations based on the cognitive-aesthetic paradigm // *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta* = *Proceedings of the Southwest State University*. 2025; 29(2): 146-165 (In Russ.). <https://doi.org/10.21869/2223-1560-2025-29-2-146-165>.

Received 02.04.2025

Accepted 29.05.2025

Published 23.07.2025

Введение

В современном информационном обществе, характеризующемся экспоненциальным ростом объёмов данных и ускорением темпов коммуникации, эффективная визуализация информации приобретает критическое значение. Презентации, будучи одним из наиболее распространённых инструментов коммуникации, играют ключевую роль в образовании, бизнесе, науке и многих других сферах деятельности. Однако создание качественных презентаций, способных удерживать внимание аудитории и эффективно доносить ключевые идеи, требует значительных временных затрат и специальных навыков в области дизайна и коммуникации. Эта проблема особенно актуальна в условиях современного динамичного мира, где время является одним из наиболее ценных ресурсов.

В связи с этим наблюдается растущий интерес к разработке инструментов, позволяющих автоматизировать процесс создания презентаций. Существующие решения в этой области предлагают различный функционал, от простого создания слайдов по шаблонам до более сложных систем, использующих алгоритмы искусственного интеллекта. Однако большинство из них сосредоточены преимущественно на технических аспектах вёрстки и форматирования, не уделяя должного внимания когнитивным и эстетическим

аспектам восприятия информации. Это приводит к тому, что автоматически сгенерированные презентации часто оказываются недостаточно эффективными с точки зрения коммуникативного воздействия на аудиторию. Они могут быть перегружены информацией, иметь негармоничную цветовую гамму или не учитывать особенности восприятия визуальной информации человеком.

В данной работе представлен инновационный подход к автоматизированной генерации презентаций, основанный на когнитивно-эстетической парадигме. Этот подход предполагает интеграцию принципов когнитивной психологии и эстетики в алгоритмы генерации визуального контента. Целью исследования является разработка программного решения, способного автоматически создавать презентации, которые были бы не только информативными, но и визуально привлекательными, учитывая особенности восприятия информации человеком.

Материалы и методы

Представленное программное решение для автоматизированного создания презентаций реализовано на языке Python с использованием объектно-ориентированного подхода [1, 2, 3, 4, 5]. Центральным элементом архитектуры является класс «CreatePresentation», инкапсулирующий основную логику создания и управления презентацией (рис.1).

```

class CreatePresentation():
    """
    Класс для создания презентации.

    Args:
        width (int): ширина презентации в Inches.
        height (int): высота презентации в Inches.
        main_color (tuple): rgb главный цвет всей презентации.

    """
    def __init__(self, width: int = 16, height: int = 9, main_color: tuple = (0, 0, 0)):
        self.prs = Presentation()
        self.width = width
        self.height = height
        self.prs.slide_width = Inches(width)
        self.prs.slide_height = Inches(height)
        self.colors = generate_design_palette(main_color, "tetradic")

```

Рис. 1. Класс «CreatePresentation»

Fig. 1. The "CreatePresentation" class

Класс «CreatePresentation» инициализируется с параметрами, определяющими размеры слайдов (ширина и высота в дюймах) и основной цвет презентации в формате RGB¹ [6]. В конструкторе класса создаётся объект презентации с помощью библиотеки «pptx» и генерируется цветовая палитра на основе заданного основного цвета и выбранного типа акцентных цветов (по умолчанию используется тетрадная палитра). Генерация цветовой палитры осуществляется отдельно реализованной функцией «generate_design_palette» [7, 8, 9].

Основную функциональность по генерации слайдов обеспечивает метод «create_any_slide» (рис. 2). Этот метод принимает в качестве аргументов тип слайда (например, main, text, image, column), содержание слайда (заголовки, тексты, пути к изображениям) и параметры форматирования (шрифты, цвета, вы-

равнивание) [10, 11]. Внутри метода «create_any_slide» осуществляется выбор соответствующего метода для генерации конкретного типа слайда. Например, для создания главного слайда вызывается метод «add_main_slide», для создания текстового слайда — метод «add_text_slide» и т.д. Такая структура обеспечивает гибкость и расширяемость системы, позволяя легко добавлять новые типы слайдов без изменения основного кода.

Каждый из методов генерации слайдов («add_main_slide», «add_text_slide», «add_image_slide», «add_columns_slide») реализует специфическую логику позиционирования и форматирования элементов на слайде. Для создания текстовых блоков используется метод «create_textbox», который позволяет устанавливать текст, шрифт, цвет, размер и выравнивание текста. Для добавления изображений используется метод «create_image», который загружает изображение из указанного файла и размещает его на слайде.

¹ Документация Python-pptx. URL: <https://python-pptx.readthedocs.io/>

```

136 elif type_slide == 'text':
137     if background_color == (-1, -1, -1):
138         background_color = self.colors['base_dark']
139     if title_color == (-1, -1, -1):
140         title_color = self.adjust_color(self.black_or_white(background_color), 0.8)
141     if main_color == (-1, -1, -1):
142         main_color = self.black_or_white(background_color)
143     self.add_text_slide(
144         title_text,
145         main_text,
146         title_font,
147         main_font,
148         title_color,
149         main_color,
150         background_color
151     )
152
153 elif type_slide == 'image':
154     if background_color == (-1, -1, -1):
155         background_color = self.colors['base_light']
156     if title_color == (-1, -1, -1):
157         title_color = self.adjust_color(self.black_or_white(background_color), 0.8)
158     if main_color == (-1, -1, -1):
159         main_color = self.black_or_white(background_color)
160     self.add_image_slide(
161         title_text,
162         main_text,
163         title_font,
164         main_font,
165         title_color,
166         main_color,
167         background_color,
168         image_path
169     )
170
171 elif type_slide == 'column':
172     if background_color == (-1, -1, -1):
173         background_color = self.colors['base_dark']
174     if title_color == (-1, -1, -1):
175         title_color = self.adjust_color(self.black_or_white(background_color), 0.8)
176     if subtitle_color == (-1, -1, -1):
177         subtitle_color = self.black_or_white(background_color)
178     if main_color == (-1, -1, -1):
179         main_color = self.black_or_white(background_color)
180     self.add_columns_slide(
181         title_text,
182         name_column_left,
183         name_column_right,
184         main_text_left,
185         main_text_right,
186         title_font,
187         subtitle_font,
188         main_font,
189         title_color,
190         subtitle_color,
191         main_color,
192         background_color
193     )
194
195 else:
196     print('!!! Такого типа слайда не существует !!!')
197     print("Возможные слайды:", *types)
198
199

```

Рис. 2. Метод «create_any_slide»

Fig. 2. The "create_any_slide" method

```

288 def add_image_slide(self, title_text: str="Тут что-то должно быть", main_text: str="Тут что-то должно быть", title_font: str="Mont Regular", main_font: str="
289 "Mont Regular", title_color: tuple=(255, 255, 255), main_color: tuple=(255, 255, 255), background_color: tuple=(0, 0, 0), image_path: str="/"):
290     """
291     Создаёт слайд с заголовком, текстом и картинкой, а также определяет соотношение сторон изображения для его удобного размещения на слайде.
292
293     Args:
294         title_text (str): текст заголовка.
295         main_text (str): текст главного содержания.
296         title_font (str): название используемого шрифта, который должен быть установлен в системе и в папке fonts.
297         main_font (str): название используемого шрифта, который должен быть установлен в системе и в папке fonts.
298         title_color (tuple): цвет (r: int, g: int, b: int).
299         main_color (tuple): цвет (r: int, g: int, b: int).
300         background_color (tuple): цвет (r: int, g: int, b: int).
301         image_path (str): адрес изображения.
302
303     """
304     if urlparse(image_path).scheme in ("http", "https"):
305         try:
306             response = requests.get(image_path, stream=True)
307             img = Image.open(io.BytesIO(response.content))
308             image_path = io.BytesIO(response.content)
309         except:
310             print("!!! ОШИБКА ПРИ ОТКРЫТИИ http ИЗОБРАЖЕНИЯ !!!")
311             print(image_path)
312
313     elif os.path.exists(image_path):
314         img = Image.open(image_path)
315     else:
316         print('!!! ИЗОБРАЖЕНИЯ НЕ СУЩЕСТВУЕТ !!!')
317         image_path = random.choice([os.path.join('mockup', 'NotFound_01.jpg'), os.path.join('mockup', 'NotFound_02.jpg'), os.path.join('mockup', 'NotFound_03.jpg')])
318         img = Image.open(image_path)
319
320     image_position_h = random.choice(['LEFT', 'RIGHT'])
321     image_position_v = random.choice(['TOP', 'BOTTOM'])
322     if img.width / img.height <= 0.8:
323         self.add_vertical_image_slide(title_text, main_text, title_font, main_font, title_color, main_color, background_color, image_path, image_position_h)
324     elif img.width / img.height >= 3.5:
325         self.add_horizontal_image_slide(title_text, main_text, title_font, main_font, title_color, main_color, background_color, image_path, image_position_v)
326     else:
327         if img.width / img.height < 1.3:
328             self.add_other_image_slide(title_text, main_text, title_font, main_font, title_color, main_color, background_color, image_path, image_position_h)
329         else:
330             self.add_other_image_slide(title_text, main_text, title_font, main_font, title_color, main_color, background_color, image_path, image_position_v)
331

```

Рис. 3. Метод «add_image_slide»

Fig. 3. The "add_image_slide" method

Особое внимание в архитектуре программного решения уделено адаптивности и динамическому расчёту параметров

элементов слайда. Например, метод «add_image_slide» анализирует соотношение сторон изображения и выбирает оп-

тимальное расположение и размер изображения на слайде, используя вспомогательные методы «`add_vertical_image_slide`», «`add_horizontal_image_slide`» и «`add_other_image_slide`». Это позволяет гармонично вписывать изображения различных форматов в общую композицию слайда.

В методе `add_image_slide` реализован алгоритм выбора оптимального расположения изображения на слайде в зависимости от его соотношения сторон. Алгоритм основан на анализе отношения ширины изображения к его высоте (`img.width / img.height`). Если это отношение меньше или равно 0.8, изображение считается вертикально-ориентированным, и вызывается метод `add_vertical_image_slide`. Если отношение больше или равно 3.5, изображение считается горизонтально-ориентированным, и вызывается метод `add_horizontal_image_slide`. В противном случае, если отношение находится в диапазоне от 0.8 до 1.3, то вызывается метод `add_other_image_slide` с положением 'LEFT' или 'RIGHT', иначе 'TOP' или 'BOTTOM'.

Каждый из методов (`add_vertical_image_slide`, `add_horizontal_image_slide`, `add_other_image_slide`) жёстко задаёт расположение элементов на слайде. Например, в методе `add_vertical_image_slide` изображение размещается либо в левой, либо в правой части слайда (в зависимости от случайного выбора `image_position_h`), а заголовок и текст

размещаются в противоположной части. Метод `add_horizontal_image_slide` размещает изображение в верхней или нижней части слайда (в зависимости от случайного выбора `image_position_v`), а заголовок и текст – в оставшейся части. Метод `add_other_image_slide` размещает изображение слева, справа, сверху или снизу, а текст с заголовком подстраиваются под него.

Следует отметить, что данный алгоритм является эвристическим и не учитывает семантику изображения и текста. В будущих версиях планируется усовершенствовать алгоритм, добавив анализ содержимого изображения и текста для более интеллектуального размещения элементов.

В случае отсутствия указанного изображения предусмотрена обработка исключений и подстановка изображения-заглушки из папки «/москир». Такой подход повышает робастность системы и предотвращает возникновение ошибок при работе с некорректными данными¹.

Метод «`create_textbox`» реализует автоматическую подгонку размера шрифта текстового блока для обеспечения оптимальной читаемости текста. Алгоритм подгонки учитывает длину текста, размер текстового блока и максимально допустимый размер шрифта. Для работы со шрифтами используется библиотека «`fontTools`», позволяющая полу-

¹ Документация FontTools. URL: <https://fonttools.readthedocs.io/en/latest/>

читать информацию о шрифте из файла «.ttf». Функция «get_font_name» извлекает имя шрифта из файла, что позволяет корректно работать с различными начертаниями шрифтов. В случае возникновения ошибок при подгонке

размера шрифта, например, при переполнении текстового блока, предусмотрена обработка исключений и установка фиксированного размера шрифта, что гарантирует отображение текста на слайде.

```

692 def adjust_color(self, rgb_color: tuple, factor=0.5) -> tuple:
693     """
694     Автоматически осветляет или затемняет цвет в зависимости от его яркости.
695
696     Args:
697         rgb_color (tuple): цвет (r: int, g: int, b: int).
698         factor: коэффициент изменения (0.0 - максимум изменения, 1.0 - без изменений).
699
700     Returns:
701         tuple: изменённый цвет (r: int, g: int, b: int).
702     """
703     brightness = 0.299 * rgb_color[0] + 0.587 * rgb_color[1] + 0.114 * rgb_color[2]
704     hls = colorsys.rgb_to_hls([x / 255.0 for x in rgb_color])
705
706     if brightness > 128:
707         hls = (hls[0], hls[1] * factor, hls[2])
708     else:
709         factor = 1 - factor
710         hls = (hls[0], 1 - (1 - hls[1]) * (1 - factor), hls[2])
711
712     return tuple(int(round(x * 255.0)) for x in colorsys.hls_to_rgb(*hls))
713
714 def black_or_white(self, color: tuple) -> tuple:
715     """
716     Выбирает белый или чёрный цвет в контраст входного.
717
718     Args:
719         color (tuple): цвет (r: int, g: int, b: int).
720
721     Returns:
722         tuple: цвет (r: int, g: int, b: int).
723     """
724
725     r, g, b = color
726     brightness = 0.299 * r + 0.587 * g + 0.114 * b
727
728     if brightness > 160:
729         return (0, 0, 0)
730     else:
731         return (255, 255, 255)
732

```

Рис. 4. Методы «adjust_color» и «black_or_white»

Fig. 4. The "adjust_color" and "black_or_white" methods

Для управления цветовой гаммой презентации используются методы «adjust_color» и «black_or_white» (рис. 4). Таким образом, метод «adjust_color» позволяет осветлять или затемнять цвет, что позволяет создавать различные оттенки на основе базовых цветов палитры. Функция «black_or_white» автоматически выбирает чёрный или белый цвет текста

в зависимости от яркости фона, обеспечивая оптимальный контраст и читаемость. Использование этих функций позволяет автоматически создавать гармоничные цветовые сочетания и адаптировать цветовую гамму к различному содержанию слайдов [12].

Для генерации цветовой палитры используется функция generate_design_palette,

основанная на цветовой модели HLS (Hue, Lightness, Saturation). Функция принимает на вход основной цвет в формате RGB и тип акцентной палитры (строка).

Основной цвет конвертируется из RGB в HLS. Создается словарь `palette`, в котором сохраняются базовые оттенки основного цвета:

- `base_light`: осветлённая версия основного цвета;
- `base_medium`: исходный основной цвет;
- `base_dark`: затемнённая версия основного цвета.

В зависимости от выбранного типа акцентной палитры (`accent_type`) генерируются дополнительные акцентные цвета:

- `complementary`: комплементарный цвет (противоположный на цветовом круге);
- `analogous`: два аналогичных цвета (соседние на цветовом круге);
- `triadic`: триадные цвета (равноудалённые на цветовом круге);
- `split_complementary`: расщеплённо-комплементарные цвета;
- `tetradic`: тетрадные цвета (четыре цвета, образующие прямоугольник на цветовом круге);
- `monochromatic`: монохроматические оттенки основного цвета.

Для каждого акцентного цвета создаются три оттенка: светлый, средний и тёмный. Сгенерированная палитра возвращается в виде словаря. Данный алгоритм обеспечивает генерацию гармоничных цветовых палитр, основан-

ных на теории цвета. Выбор различных типов акцентных палитр позволяет создавать разнообразные визуальные решения для презентаций.

В целом, представленная архитектура программного решения демонстрирует продуманный и эффективный подход к автоматизированной генерации презентаций. Использование объектно-ориентированного подхода, адаптивных алгоритмов и внешних библиотек обеспечивает гибкость, расширяемость и надёжность системы.

Интеграции моделей ИИ (LLM и ТТИ)

В рамках предложенного решения, центральным звеном которого является модуль для автоматизированной генерации презентаций, реализована интеграция с моделями искусственного интеллекта, а именно с большими языковыми моделями (LLM) и моделями преобразования текста в изображение (ТТИ). Эта интеграция позволяет автоматизировать процесс создания контента для презентаций, делая его более гибким и адаптивным к различным задачам.

В общем виде система функционирует следующим образом: на вход модуля поступают определённые данные, которые могут включать в себя текстовую информацию, а также, опционально, изображения. Далее, в зависимости от настроек, модуль инициирует взаимодействие с одной или несколькими LLM, такими как Claude или ChatGPT. Взаимодействие осуществляется посредством API, при этом модуль формирует

запросы к LLM, передавая им собранные исходные данные и запрашивая генерацию текстового контента, релевантного поставленной задаче.

Важно отметить, что формат запросов и структура получаемых от LLM данных могут варьироваться в зависимости от конкретной задачи и используемой языковой модели.

Полученный текстовый контент далее анализируется и, при необходимости, дополняется визуальными элементами. Для генерации изображений используются ТГГ модели, например, DALL-E или Flux.

Модуль формирует текстовые промпты для ТГГ моделей на основе информации, полученной от LLM, а также, при наличии, на основе анализа исходных изображений, если таковые были предоставлены. Сгенерированные ТГГ изображения, наряду с текстовым контентом от LLM, становятся основой для формирования слайдов презентации.

При этом модуль оперирует абстракциями, позволяющими создавать слайды различного типа, наполняя их текстом, изображениями, а также другими элементами, такими как таблицы или диаграммы.

Важной особенностью является алгоритм автоматической компоновки элементов на слайде, который учитывает тип и размер добавляемых объектов, стремясь к оптимальному визуальному расположению.

Ключевым аспектом интеграции является гибкость системы, позволяющая использовать различные LLM и ТГГ мо-

дели, а также адаптировать процесс генерации контента под разнообразные задачи, не ограничиваясь одним возможным решением, а распространяясь на создание презентаций любой тематики, где требуется автоматизированная генерация текстового и визуального контента на основе входных данных.

Модуль предоставляет возможность параметризации процесса создания презентации, позволяя управлять различными аспектами, такими как выбор используемых моделей ИИ, стилизация, форматирование текста, что делает его мощным инструментом для широкого круга задач.

Обоснование выбора технологий

Выбор технологий является важным этапом разработки любого программного решения. Для данного проекта мы руководствовались рядом критериев, включая простоту использования, наличие необходимых функциональных возможностей, производительность, поддержку сообщества и возможность интеграции с другими технологиями.

В качестве основного языка программирования был выбран Python. Python является одним из самых популярных языков, особенно в области искусственного интеллекта, благодаря своему простому синтаксису, большому количеству библиотек и обширному сообществу. Кроме того, Python обладает отличной совместимостью с библиотеками для машинного обучения, что критически важно для интеграции с моделями LLM и ТГГ.

Для создания и редактирования презентаций была выбрана библиотека `pptx.pptx` – это одна из наиболее популярных и функциональных библиотек Python для работы с файлами PowerPoint. Она позволяет создавать презентации с нуля, добавляя различные элементы и форматируя их. `.pptx` имеет подробную документацию, активно развивается и поддерживается сообществом.

Помимо `pptx` существуют и другие библиотеки для работы с презентациями на Python, такие как `python-pptx-interface`, `ReportLab` и `Aspose.Slides`:

- `python-pptx-interface` является над-стройкой над `pptx` и предоставляет более высокоуровневый интерфейс, но обладает ограниченной функциональностью;

- `ReportLab` – это мощная библиотека для создания PDF-документов, которую можно использовать и для генерации презентаций, но она требует больше усилий для настройки и не ориентирована на формат PowerPoint из-за чего теряется адаптивность, которая доступна в `python-pptx`;

- `Aspose.Slides` – это кроссплатформенная коммерческая библиотека с широкими и удобными возможностями, но она является платной с высокой стоимостью и подписочной системой.

Стоит отметить, что `pptx` имеет и некоторые ограничения. Например, она не предоставляет полного контроля над всеми аспектами форматирования, как это можно сделать вручную в PowerPoint. Однако эти ограничения не являются кри-

тичными для данного проекта. Преимущества же `pptx`, такие как простота использования и широкие функциональные возможности, делают её оптимальным выбором для реализации поставленной задачи.

Реализация когнитивных принципов

Одним из ключевых когнитивных принципов, реализованных в нашем решении, является принцип когнитивной нагрузки. Согласно теории когнитивной нагрузки, рабочая память человека имеет ограниченную ёмкость. Перегрузка рабочей памяти избыточной информацией приводит к снижению эффективности обучения и запоминания. Для минимизации когнитивной нагрузки мы используем несколько подходов. Во-первых, метод автоматически подгоняет размер шрифта, ограничивая количество текста на слайде и обеспечивая его читаемость. Во-вторых, алгоритмы динамического расчета параметров изображений, реализованные в методах `add_vertical_image_slide`, `add_horizontal_image_slide` и `add_other_image_slide`, оптимизируют размер и расположение визуальных элементов, предотвращая перегрузку зрительного канала.

Для облегчения восприятия информации и установления логических связей между элементами на слайде мы используем принцип близости, один из гештальт-принципов зрительного восприятия. Этот принцип гласит, что объекты, расположенные близко друг к дру-

гу, воспринимаются как единая группа. В нашем решении принцип близости реализуется через алгоритмы автоматического макетирования, которые группируют связанные текстовые блоки, заголовки и изображения. Например, при использовании метода `add_columns_slide` система автоматически располагает текстовые колонки на оптимальном расстоянии друг от друга, а подписи к изображениям размещаются непосредственно под ними, создавая чёткое визуальное единство. Это позволяет аудитории интуитивно понимать взаимосвязь между различными элементами слайда.

Чтобы акцентировать внимание аудитории на наиболее важной информации, мы применяем принцип выделения. Этот принцип предполагает использование визуальных подсказок для направления внимания на ключевые элементы презентации. В нашем программном решении принцип выделения реализуется через методы управления цветом и шрифтом. Например, метод `adjust_color` позволяет создавать различные цветовые акценты на основе базовой палитры, выделяя заголовки, ключевые слова или важные графические элементы. Функция `black_or_white` автоматически выбирает контрастный цвет текста в зависимости от фона, обеспечивая читаемость и привлекая внимание к текстовому контенту. Кроме того, метод `create_textbox` позволяет варьировать размер и начертание шрифта, выделяя заголовки и подзаголовки. Эти инструменты позволяют создавать визуальную

иерархию на слайде, направляя внимание аудитории по заранее определённой траектории.

Результаты и их обсуждение

В настоящее время на рынке представлен ряд инструментов, предлагающих возможности автоматизированной или полуавтоматизированной генерации презентаций. Среди наиболее известных можно выделить Canva, PowerPoint Designer (в составе Microsoft PowerPoint), Google Slides (с функцией "Умный подбор макета"), Gamma. Целью данного раздела является проведение детального сравнительного анализа предложенного решения с существующими аналогами, что позволит выявить его сильные и слабые стороны, а также определить его конкурентные преимущества.

В табл. 1 представлена сравнительная характеристика существующих методов автоматизированной генерации презентаций и предлагаемого решения на основе когнитивно-эстетической парадигмы, где цифры от 1 до 5 в таблице представляют собой оценочную шкалу, где каждая цифра соответствует определённому уровню качества, функциональности или характеристик по каждому критерию. Шкала интерпретируется следующим образом:

- 1 – Очень низкий уровень;
- 2 – Низкий уровень;
- 3 – Средний уровень;
- 4 – Высокий уровень;
- 5 – Очень высокий уровень.

Таблица 1. Сравнительная характеристика существующих методов генерации презентаций и предлагаемого подхода**Table 1.** Comparative characteristics of existing presentation generation methods and the proposed approach

Критерий / Criteria	Canva	PowerPoint Designer	Google Slides	Gamma	Предлагаемое решение на основе когнитивно-эстетической парадигмы / The proposed solution is based on the cognitive-aesthetic paradigm
Степень автоматизации	3	3	2	5	5
Качество дизайна	5	3	3	5	5
Настраиваемость	5	3	5	4	3
Удобство использования	4	4	3	4	5
Поддержка разных типов контента	5	5	5	4	5
Интеграция с другими сервисами	3	5	5	4	3
Стоимость (больше = дороже)	5	5	3	5	1
Учёт когнитивных принципов	5	3	2	3	5
Учёт эстетических принципов	5	3	2	4	5
Доступность в РФ	Нет	Нет	Нет	Нет	Да

Анализ таблицы показывает, что предложенное решение превосходит существующие аналоги в некоторых аспектах. Canva, являясь мощным и удобным инструментом для ручного дизайна, значительно уступает в степени автоматизации и интеграции с другими сервисами, т.к. работает онлайн и некоторые шаблоны поддерживают исключительно официальный сайт Canva.

PowerPoint Designer и Google Slides предлагают определённые функции автоматизации, но они ограничены рамками шаблонов и не обеспечивают комплексного подхода к генерации контента.

Gamma выделяется высокой степенью автоматизации и является отличным решением для быстрого создания презентаций, но за этот инструмент придётся не мало заплатить.

Предложенное решение, благодаря интеграции LLM и TTI моделей, а также использованию алгоритмов, учитывающих принципы дизайна и когнитивной психологии, позволяет создавать презентации, которые не только эстетически привлекательны, но и эффективны с точки зрения восприятия и усвоения информации. По времени создания и стоимости предложенное решение также демонстрирует преимущество, позволяя генерировать презентации быстрее, чем при использовании некоторых аналогов, но при этом затрачивая меньше ресурсов.

Таким образом, предлагаемое решение для автоматизированной генерации презентаций представляет значительные преимущества перед существующими методами. Оно не только обеспечивает высококачественное создание слайдов с учетом когнитивных и эстетических аспектов, но и предлагает гибкость, адаптивность и удобство использования. Это позволяет создавать презентации, которые не просто информируют, но и удерживают внимание аудитории, а также эффективно передают ключевые идеи.

Экспериментальная проверка и оценка пользовательского опыта

Для оценки эффективности разработанного решения было проведено исследование с участием реальных пользователей. В исследовании приняли участие 50 человек, разделённых на две группы:

1. Группа А: 25 пользователей, которые создавали презентации с использованием разработанного инструмента.

2. Группа В: 25 пользователей, которые создавали презентации вручную с помощью PowerPoint.

Участникам обеих групп было предложено создать презентацию на заданную тему за ограниченное время (30 минут). После завершения задачи участники оценивали свои презентации по следующим критериям: время, затраченное на создание презентации; удобство использования инструмента; качество дизайна.

Для оценки эффективности восприятия информации презентации, созданные обеими группами, были показаны независимой аудитории из 100 человек. Аудитория оценивала презентации по следующим параметрам: чёткость изложения информации; визуальная привлекательность; способность удерживать внимание; общее впечатление.

Результаты исследования показаны в табл. 2.

Результаты исследования подтвердили, что разработанное решение не только упрощает процесс создания презентаций, но и обеспечивает более высокое качество визуального контента.

Эффективность восприятия информации, оцениваемая независимой аудиторией, продемонстрировала статистически значимое превосходство презентаций, созданных с использованием разработанного инструмента (группа А), по сравнению с презентациями, сгенерированными вручную с применением тра-

диционных методов (группа В). Результаты оценки по ключевым параметрам представлены ниже:

Чёткость изложения информации: презентации группы А получили средний балл 4.8 из 5, что свидетельствует о высокой степени структурированности и логической связности представленного контента. В то же время презентации группы В были оценены на 4.2 из 5, что указывает на наличие определённых трудностей в восприятии информации,

связанных с недостаточной оптимизацией компоновки и форматирования.

Средняя оценка визуальной привлекательности презентаций группы А составила 4.7 из 5, что подтверждает эффективность применения когнитивно-эстетической парадигмы в автоматизированной генерации дизайна. Для группы В данный показатель составил 4.0 из 5, что может быть связано с отсутствием системного подхода к выбору цветовых решений, шрифтов и композиции.

Таблица 2. Сравнительная характеристика существующих методов генерации презентаций и предлагаемого подхода

Table 2. Comparative characteristics of existing presentation generation methods and the proposed approach

	Время, затраченное на создание презентации / Time spent on creating	Удобство использования инструмента / Ease of use of the tool	Качество дизайна / a presentation Design quality
Участники группы А	15	4,6	4,7
Участники группы В	25	3,8	4,1

Презентации группы А были оценены на 4.6 из 5, что подчёркивает их способность эффективно привлекать и поддерживать интерес аудитории благодаря оптимальному сочетанию визуальных и текстовых элементов. В группе В данный параметр получил оценку 4.1 из 5, что указывает на возможную перегруженность слайдов информацией или недостаточную визуальную гармонию.

Таким образом, результаты исследования подтверждают, что разработанное решение обеспечивает более высокий уровень восприятия информации

за счёт интеграции когнитивных и эстетических принципов в процесс автоматизированной генерации презентаций. Это позволяет не только улучшить визуальное качество контента, но и повысить его коммуникативную эффективность, что является ключевым преимуществом предложенного подхода.

Перспективы интеграции с ИИ

Разработанный модуль для создания презентации может быть использован в процессе генерации презентации, которая может базироваться на конвейерной ар-

хитектуре, интегрирующей несколько моделей искусственного интеллекта.

Вначале большая языковая модель (LLM) выполняет семантический анализ определённых входных данных [13, 14, 15]. Далее, осуществляется внутренний алгоритм для создания нарратива, который может быть декомпозирован для генерации индивидуального слайда презентации (с помощью ключевых слов триггеров).

Параллельно с этим, модель преобразования текста в изображение (ТТИ) использует сгенерированный LLM текст в качестве промптов для синтеза

визуального контента [16, 17, 18]. Сгенерированные изображения оптимизируются по разрешению и композиции, соответствующим выбранному шаблону презентации.

Далее, модуль для создания презентаций обрабатывает текстовые и визуальные артефакты, созданные LLM и ТТИ. Реализуются алгоритмы автоматического макетирования и стилизации, адаптируя расположение элементов на слайде. Пример сгенерированной презентации с помощью модуля и ИИ можно увидеть на рис. 5-6.

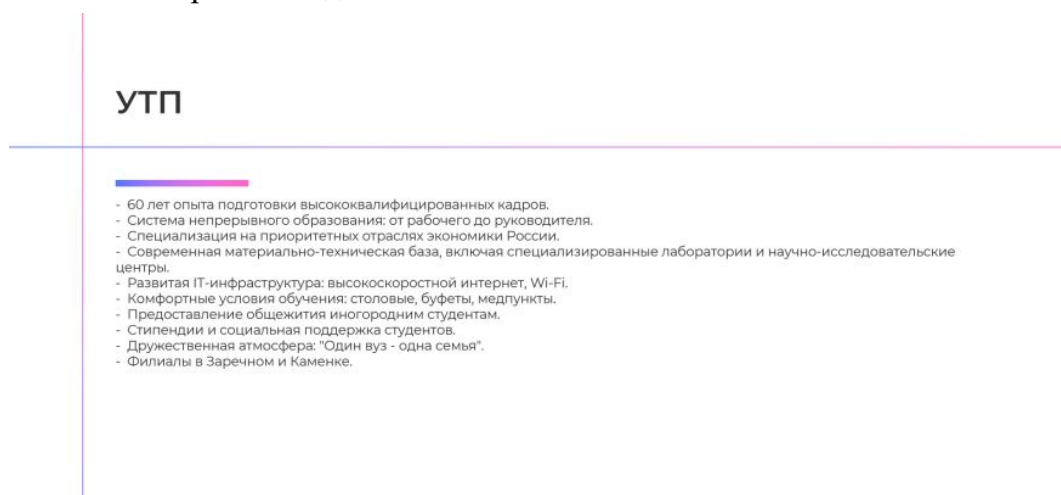


Рис. 5. Пример сгенерированной презентации с помощью разработанного модуля и ИИ

Fig. 5. An example of a generated presentation using the developed module and AI

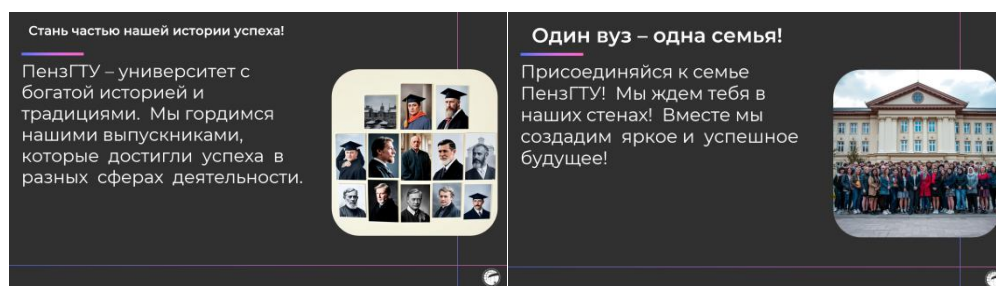


Рис. 6. Пример сгенерированной презентации с помощью разработанного модуля и ИИ

Fig. 6. An example of a generated presentation using the developed module and AI

Вся процедура может быть автоматизирована и управляема главным скриптом, который координирует взаимодействие между различными модулями и моделями ИИ. Эта архитектура позволяет эффективно генерировать высококачественные презентации, минимизируя необходимость ручного вмешательства.

Дизайнерские решения

Для фона главных слайдов был выбран тёмно-серый цвет, определённый в качестве основного цвета презентации. Этот выбор обеспечивает высокий контраст с белым цветом текста и элементов интерфейса, улучшая читаемость. В качестве акцентных цветов используются оттенки синего и фиолетового, полученные с помощью функции `generate_design_palette` с типом акцентной палитры `tetradic`. Эти цвета применяются в градиентных линиях и кругах, добавляя глубины и динамики фону.

Для заголовка используется шрифт `Montserrat SemiBold`, обеспечивающий акцентирование и хорошую читаемость. Для основного текста - `Montserrat Regular`, создающий контраст с заголовком и сохраняющий иерархию.

Логотип (если он указан) помещается в правой нижней части слайда, занимая место после пересечения двух градиентных линий, создавая единообразный стиль и визуально уравнивая слайд.

Изображения и текст на слайдах занимают столько места, чтобы не пере-

грузить человека информацией для лучшего усвоения материала.

Таким образом, соблюдая все аспекты визуального восприятия, создаётся лёгкая в чтении и усвоении презентация.

В целом, автоматическая генерация демонстрирует хорошие результаты в создании стандартных презентаций, где требуется соблюдение базовых правил дизайна и быстрое получение результата (несколько минут). Однако для проектов, требующих высокой степени креативности и оригинальности, по-прежнему предпочтительнее привлечение опытного дизайнера.

Вывод

В данной работе представлен подход к автоматизированной генерации презентаций, основанный на когнитивно-эстетической парадигме. Разработано программное решение, которое не только упрощает процесс создания визуального контента, но и учитывает когнитивные и эстетические аспекты восприятия информации.

Анализ существующих решений показал, что многие из них ограничены в своих возможностях и не способны эффективно адаптироваться к потребностям пользователей [19, 20]. В то время как шаблонные системы и инструменты на основе искусственного интеллекта обеспечивают автоматизацию и упрощение вёрстки, они часто пренебрегают важными факторами, такими как гармония визуального оформления, оптимальное восприятие инфор-

мации и удержание внимания аудитории [21]. Эти недостатки подчеркивают необходимость применения более комплексного подхода, который был реализован в нашем решении.

Разработанная архитектура программного обеспечения, основанная на объектно-ориентированном подходе, демонстрирует возможности для гибкой и адаптивной генерации слайдов. Использование специализированных методов для обработки текста и изображений позволяет создать гармоничные и чита-

бельные презентации. Алгоритмы не только учитывают содержание, но и оптимизируют визуальные параметры, чтобы улучшить восприятие информации. В результате, предлагаемое решение существенно превосходит существующие методы в вопросах создания презентаций, предоставляя пользователю инструменты для генерации высококачественного контента, который эффективно передаёт ключевые идеи и поддерживает интерес аудитории.

Список литературы

1. Златопольский Д.М. Основы программирования на языке Python. М.: ДМК Пресс, 2017. 284 с.
2. Свейгарт Эл. Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих. М.: Вильямс, 2016. 592 с.
3. Гэддис Т. Начинаем программировать на Python. 4-е изд. СПб.: БХВ-Петербург, 2019. 768 с.
4. Федоров Д. Ю. Программирование на языке высокого уровня Python. М.: Изд-во Юрайт, 2019. 161 с. URL: <https://urait.ru/bcode/437489>
5. Рейтц К., Шлюссер Т. Автостопом по Python. СПб.: Питер, 2017. 336 с. (Серия «Бестселлеры O'Reilly»).
6. Буч Г. Объектно-ориентированный анализ и проектирование на примерах приложений. М.: ДМК Пресс, 2007. 520 с.
7. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Партнер, Д. Влссидес. СПб.: Питер, 2009. 368 с.
8. Чирвинский А. Основы теории цвета в графическом дизайне // Журнал дизайна и теории цвета. 2019. № 3. С. 45–53.
9. Хайн Дж. Алгоритмы автоматической настройки цвета для приложений обработки изображений // Журнал цифровой обработки изображений. 2020. Т. 33, № 4. С. 124–130.
10. Стивенс Х. Интеллектуальная генерация презентаций в динамических средах // Международный журнал передовой компьютерной науки и приложений. 2019. Т. 10, № 6. С. 155–161.

11. Осборн А.Ф. Разработка презентаций для визуальной ясности // Журнал деловой и технической коммуникации. 2018. Т. 32, № 1. С. 87–102.
12. Смит Дж. Проектирование адаптивного пользовательского интерфейса // Труды IEEE по системам «человек-машина». 2022. Т. 52, № 2. С. 320–330.
13. Рой П., Панди С., Джайн К. Глубокое обучение для обработки естественного языка: применение глубоких нейронных сетей к задачам машинного обучения в области обработки естественного языка. Cham: Springer, 2018. 280 с.
14. Ротман Д. Трансформаторы для обработки естественного языка: создание и развертывание современных конвейеров обработки естественного языка. Бирмингем: Packt Publishing, 2021. 374 с.
15. Яричев М. GPT-3: Создание инновационных продуктов обработки естественного языка с использованием API OpenAI GPT-3. Независимое издание, 2021. 210 с.
16. Скрыбин А. Практическая генерация изображений с TensorFlow: создание генеративных моделей с использованием VAE, GAN и моделей диффузии. Бирмингем: Packt Publishing, 2022. 380 с.
17. Джавад М., Лонгхи А. Машинное обучение для визуальной аналитики данных: от моделей глубокого обучения до GAN и VAE. Cham: Springer, 2021. 420 с.
18. Фостер Д. Генеративное глубокое обучение: обучение машин рисовать, писать, сочинять и играть. Севастополь: O'Reilly Media, 2019. 352 с.
19. Wong D. M., Shankaranarayanan G. Automated Data Visualization: A Review and Future Directions // Journal of Visual Languages & Computing. 2022.
20. Allison J., McNamara S. Data Storytelling for Business: The Essential Guide to Engaging Visualization. Springer, 2021.
21. Gomez, E., Decker, S. Generative Design for Data Visualization: A Review and Framework // Computer Graphics Forum. 2022.

References

1. Zlatopolsky D.M. Fundamentals of programming in Python. Moscow: DMK Press; 2017. 284 p. (In Russ.).
2. Sveygart, El. Automation of routine tasks using Python: a practical guide for beginners. Moscow: Williams; 2016. 592 p. (In Russ.).
3. Gaddis T. Starting to program in Python. St. Petersburg: BHV-Petersburg; 2019. 768 p. (In Russ.).
4. Fedorov D. Yu. Programming in a high-level Python language: a textbook for applied bachelor's degree. Moscow: Yurait; 2019. 161 p. (In Russ.). Available at: <https://urait.ru/bcode/437489>
5. Reitz K., Schlusser T. The Hitchhiker's guide to Python. St. Petersburg: Piter; 2017. 336 p. (In Russ.).

6. Butch G. Object-oriented analysis and design using application examples. Moscow: DMK Press; 2007. 520 p. (In Russ.)
7. Gamma E., Helm R., Partner R., Vlissides D. Object-oriented design techniques. Design patterns. St. Petersburg: Peter; 2009. 368 p. (In Russ.)
8. Chirvinsky A. Fundamentals of color theory in graphic design. *Zhurnal dizaina i teorii tsveta = Journal of Design and Color Theory*. 2019; (3): 45-53. (In Russ.)
9. Hein J. Algorithms for automatic color adjustment for image processing applications. *Zhurnal tsifrovoy obrabotki izobrazhenii = Journal of Digital Image Processing*. 2020; 33 (4): 124-130. (In Russ.).
10. Stevens H. Intelligent generation of presentations in dynamic environments. *Mezhdunarodnyi zhurnal peredovoi komp'yuternoi nauki i prilozhenii = International Journal of Advanced Computer Science and Applications*. 2019; 10(6): 155-161. (In Russ.)
11. Osborn A.F. Development of presentations for visual clarity. *Zhurnal delovoi i tekhnicheskoi kommunikatsii = Journal of Business and Technical Communication*. 2018; 32 (1): 87-102. (In Russ.)
12. Smith J. Designing an adaptive user interface. *Trudy IEEE po sistemam «chelovek-mashina» = IEEE Proceedings on human-machine systems*. 2022; 52 (2): 320-330. (In Russ.)
13. Roy P., Pandi S., Jain K. Deep learning for natural language processing: the application of deep neural networks to machine learning tasks in the field of natural language processing. Cham: Springer; 2018. 280 p. (In Russ.)
14. Rotman D. Transformers for natural language processing: creation and deployment of modern natural language processing pipelines. Birmingham: Packt Publishing; 2021. 374 p. (In Russ.).
15. Yarichev M. GPT-3: Creating innovative natural language processing products using the OpenAI GPT-3 API. Independent edition, 2021. 210 p. (In Russ.)
16. Scriabin A. Practical image generation with TensorFlow: creation of generative models using VAE, GAN and diffusion models. Birmingham: Packt Publishing; 2022. 380 p. (In Russ.)
17. Javad M., Longhi A. Machine learning for visual data analytics: from deep learning models to GAN and VAE. Cham: Springer; 2021. 420 p. (In Russ.).
18. Foster D. Generative deep learning: teaching machines to draw, write, compose and play. Sevastopol: O'Reilly Media; 2019. 352 p. (In Russ.).
19. Wong D. M., Shankaranarayanan, G. Automated Data Visualization: A Review and Future Directions. Shankaranarayanan. *Journal of Visual Languages & Computing*. 2022.
20. Allison J., McNamara S. Data Storytelling for Business: The Essential Guide to Engaging Visualization. Springer; 2021.
21. Gomez E., Decker S. Generative Design for Data Visualization: A Review and Framework. *Computer Graphics Forum*. 2022.

Информация об авторах / Information about the Authors

Зоткина Алена Александровна, старший преподаватель кафедры «Программирование», Пензенский государственный технологический университет, г. Пенза, Российская Федерация, e-mail: alena.zotkina.97@mail.ru, ORCID: <http://orcid.org/0000-0002-2497-6433>

Alena A. Zotkina, Senior Lecturer of the Programming Department, Penza State Technological University, Penza, Russian Federation, e-mail: alena.zotkina.97@mail.ru, ORCID: <http://orcid.org/0000-0002-2497-6433>

Мартышкин Алексей Иванович, кандидат технических наук, доцент, заведующий кафедрой «Программирование», Пензенский государственный технологический университет, г. Пенза, Российская Федерация, e-mail: alexey314@yandex.ru, ORCID: <http://orcid.org/0000-0002-3358-4394>

Alexey I. Martyshkin, Cand of Sci. (Engineering), Associate Professor, Head of the Programming Department, Penza State Technological University, Penza, Russian Federation, e-mail: alexey314@yandex.ru, ORCID: <http://orcid.org/0000-0002-3358-4394>

Павлов Аким Алексеевич, студент кафедры «Программирование», Пензенский государственный технологический университет, г. Пенза, Российская Федерация, e-mail: akimpavlov03@yandex.ru, ORCID: <http://orcid.org/0009-0007-9646-2064>

Akim A. Pavlov, Student of the Programming Department, Penza State Technological University, Penza, Russian Federation, e-mail: akimpavlov03@yandex.ru, ORCID: <http://orcid.org/0009-0007-9646-2064>

Ткаченко Александра Васильевна, студентка кафедры «Программирование», Пензенский государственный технологический университет, г. Пенза, Российская Федерация, e-mail: Aleksandratka4@yandex.ru, ORCID: <http://orcid.org/0009-0007-8125-8167>

Alexandra V. Tkachenko, Student of the Programming Department, Penza State Technological University, Penza, Russian Federation, e-mail: Aleksandratka4@yandex.ru, ORCID: <http://orcid.org/0009-0007-8125-8167>