Федеральное государственное бюджетное образовательное учреждение высшего образования

«Новосибирский государственный технический университет» Кафедра теоретической и прикладной информатики

# ОТЧЕТ ПО ПРАКТИКЕ

Производственная практика: технологическая (проектно-технологическая) практика

(наименование практики в соответствии с учебным планом)

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

|  |  |
| --- | --- |
| Выполнил:  Студент Швадченко А. В.  *(Ф.И.О.)*  Группа ПМИ-12  Факультет ПМИ  *подпись*  « » 2024 г. | Проверил:  Руководитель от НГТУ  *(Ф.И.О.)*  Балл: , ECTS, Оценка  «отлично», «хорошо», «удовлетворительно», «неуд.»,  «зачтено», «не зачтено»  *подпись*  « » 2024 г. |

Новосибирск 2024

Оглавление

[ОТЧЕТ ПО ПРАКТИКЕ 3](#_Toc185943692)

[Введение 3](#_Toc185943693)

[Описание задачи 3](#_Toc185943694)

[Введение в предметную область 3](#_Toc185943695)

[Актуальность задачи 4](#_Toc185943696)

[Основная часть 5](#_Toc185943697)

[Описание используемых методов 5](#_Toc185943698)

[Результаты работы 7](#_Toc185943699)

[Заключение 10](#_Toc185943700)

[Литература 10](#_Toc185943701)

[Приложение 1 11](#_Toc185943702)

[Приложение 2 15](#_Toc185943703)

# Введение

## Описание задачи

Задание, которое было получено от руководителя пратики, содержало в себе изучение и выбор методик оценивания для проверки выполнения тестов студентами, уделяя особое внимание вопросам с развернутым ответом, с последующим написанием программы, а также изучение методик работы с текстовой информацией.

Данное задание подразумевает использование технологий нейронной сети Chat GPT - 4.

Также задание подразумевало использование различных интернет ресурсов или иных источников информации, которые позволили бы получить необходимые знания о написании программы.

## Введение в предметную область

В современных образовательных учреждениях проверка знаний студентов остаётся одной из ключевых задач для преподавателей. Процесс оценки студенческих работ зачастую связан с ручной проверкой, что требует значительных временных и интеллектуальных затрат. Автоматизация проверки ответов является важным шагом в развитии образовательных технологий.

С развитием машинного обучения и обработки естественного языка стало возможным автоматизировать значительную часть этой задачи. Одним из ключевых аспектов такой автоматизации является использование алгоритмов для анализа текстов ответов студентов, включая лемматизацию, то есть приведение слов к начальной форме. Это позволяет упростить поиск совпадений с правильным ответом. Кроме того, использование искусственного интеллекта, такого как языковая модель GPT, добавляет уровень интеллектуального анализа, способного оценивать качество ответа не только на основе ключевых слов, но и по смыслу.

Программа, представленная в данном отчёте, использует возможности языка Python и интеграцию с API GPT для:

* Загрузки и структурированного отображения вопросов из файлов JSON.
* Лемматизации и анализа ключевых слов в ответах студентов.
* Автоматической генерации оценок с учётом как лексических совпадений, так и с помощью нейросети Chat GPT.

Эта программа позволяет минимизировать время на проверку, увеличивает объективность оценивания и снижает нагрузку на преподавателей. Её применение особенно актуально в условиях увеличения числа студентов и необходимости оперативной обратной связи. Развитие таких систем создаёт новые возможности для образовательных учреждений, улучшая качество обучения и управления учебным процессом.

## Актуальность задачи

Актуальность задачи разработки программы для анализа и автоматизированной проверки ответов студентов с использованием ChatGPT обусловлена ростом потребности в цифровизации образовательных процессов и повышения их эффективности. Современные образовательные учреждения сталкиваются с необходимостью автоматизации рутинных задач, включая проверку текстовых ответов студентов, что традиционно занимает много времени и требует значительных ресурсов со стороны преподавателей.

Использование технологий обработки естественного языка, таких как ChatGPT, предоставляет уникальные возможности для реализации таких процессов. Это позволяет не только ускорить проверку ответов, но и повысить объективность оценки благодаря алгоритмам, анализирующим текст с учетом контекста.

Программа, построенная на интеграции с ChatGPT, способна учитывать как семантическое содержание ответа, так и ключевые слова, что делает процесс проверки более комплексным и точным.

Таким образом, разработка подобного программного обеспечения способствует решению ряда задач:

* Ускорение проверки ответов, что важно в условиях ограниченных временных ресурсов.
* Повышение объективности оценки за счет применения стандартизированных алгоритмов.
* Снижение нагрузки на преподавателей, что позволяет сосредоточиться на более сложных аспектах образовательного процесса.

Применение подобных инструментов отражает современные тренды в цифровизации образования и делает учебный процесс более эффективным и современным.

# Основная часть

## Описание используемых методов

В данной работе используются модули и библиотеки Python, которые объединены в приложение для проверки ответов студентов. Они выполняют следующие функции:

**tkinter**  
tkinter — стандартный модуль Python для создания графических пользовательских интерфейсов.  
Основные особенности:

1. Позволяет создавать окна с кнопками, текстовыми полями и другими элементами управления.
2. Поддерживает обработку событий, таких как нажатия кнопок или ввод текста.
3. Обеспечивает гибкость в настройке интерфейса, включая размер, расположение и стиль элементов.

В данном проекте tkinter используется для создания интерфейса приложения, где студенты вводят свои ответы, а преподаватель просматривает результаты проверки.

**pymorphy3**  
pymorphy3 — библиотека для морфологического анализа текста на русском языке.  
Основные возможности:

1. Лемматизация — приведение слов к их начальной форме (например, "машины" → "машина").
2. Определение частей речи для слов.
3. Учет грамматических особенностей, таких как падежи, числа и род.

В этом проекте pymorphy3 используется для обработки текста, включая лемматизацию и фильтрацию ненужных слов (например, предлогов и союзов), что помогает улучшить точность проверки ответов.

**requests**  
requests — библиотека для выполнения HTTP-запросов.  
Ключевые характеристики:

1. Упрощает отправку запросов к API и работу с ответами.
2. Поддерживает GET, POST и другие методы запросов.
3. Обрабатывает заголовки, параметры и сессии.

В данном проекте requests используется для отправки запросов к внешнему API (например, GPT-4) для оценки ответа студента.

**json**  
json — стандартный модуль Python для работы с JSON-данными.  
Особенности:

1. Преобразование данных из формата JSON в Python-объекты и обратно.
2. Удобство при работе с файлами и API.

JSON используется для загрузки вопросов и правильных ответов из файла, а также для формирования запросов к API.

**Объединение модулей**  
Созданное приложение сочетает функционал вышеуказанных библиотек:

1. tkinter предоставляет графический интерфейс для взаимодействия с пользователем.
2. pymorphy3 помогает анализировать текстовые ответы, выделяя ключевые слова.
3. requests выполняет запросы к API для оценки ответов.
4. json позволяет хранить и обрабатывать данные в формате, удобном для обмена между системами.

Это делает приложение мощным инструментом для автоматизации проверки знаний студентов.

## Результаты работы

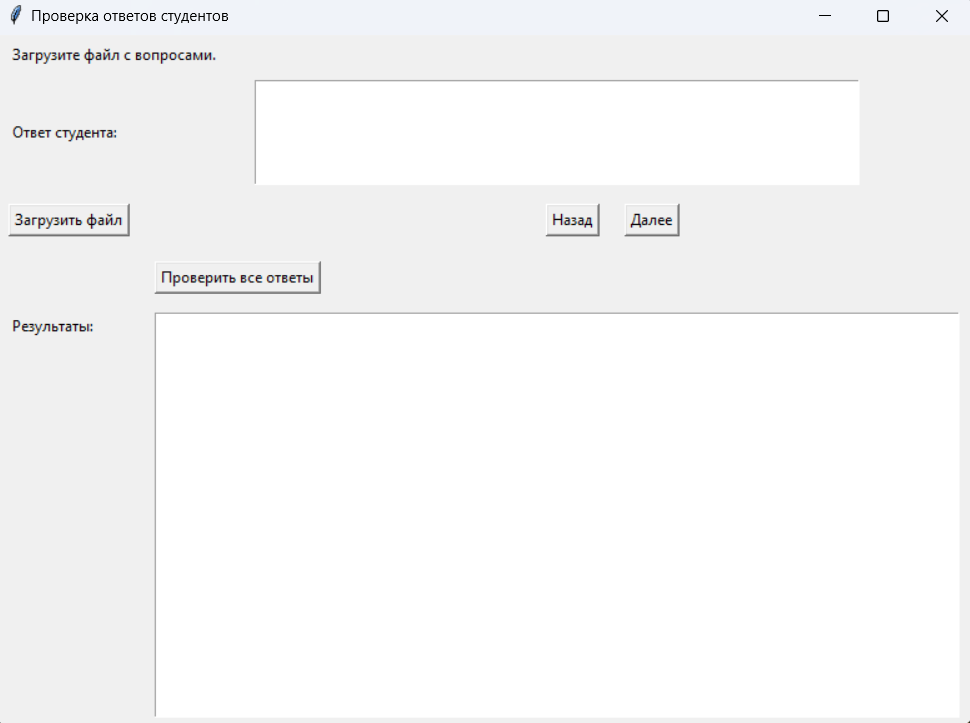


Рисунок 1 - Окно программы.

Загружаем json файл с помощью кнопки “Загрузить файл”.

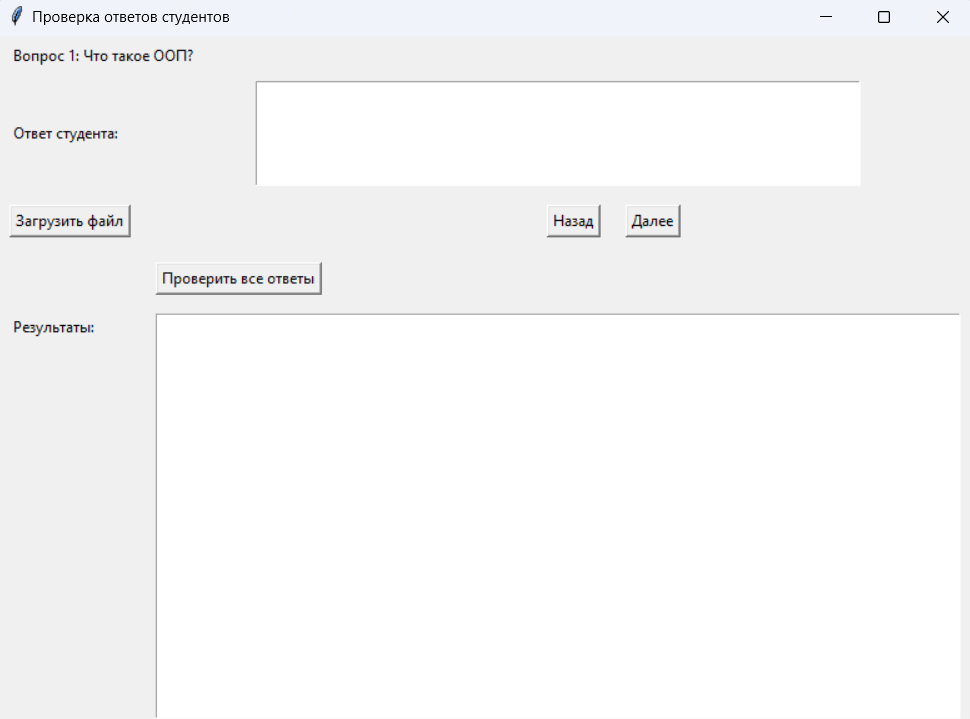


Рисунок 2

Теперь нужно внести ответы на вопросы в соответствующее поле. Между вопросами можно свободно перемещаться. После заполнения всех ответов необходимо нажать “Проверить все ответы”.

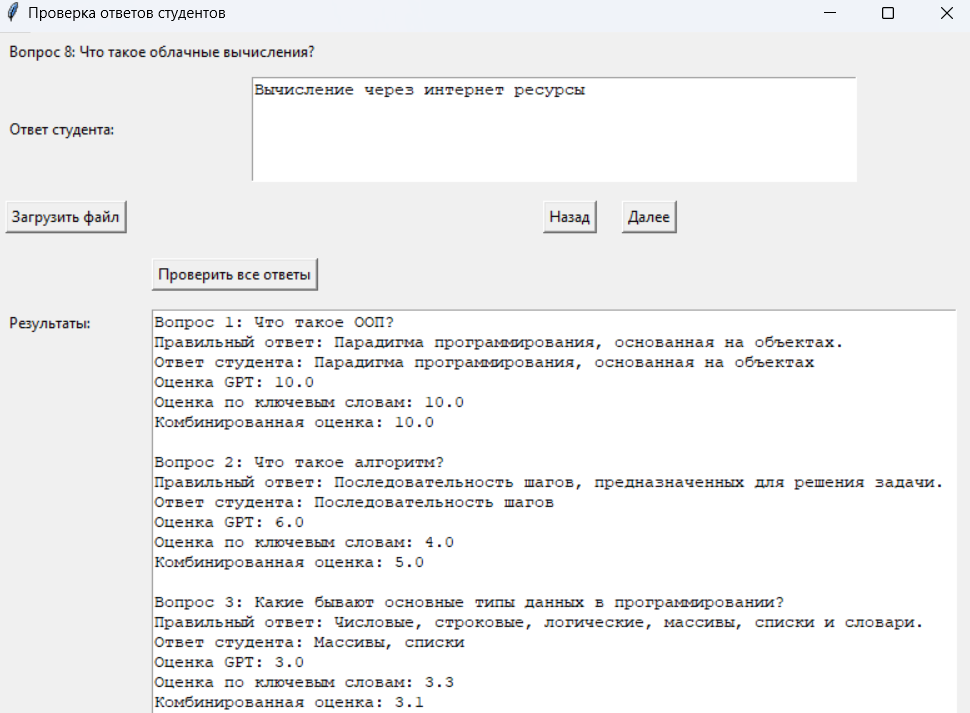


Рисунок 3 – Проверка на полностью правильном ответе и неполных ответов

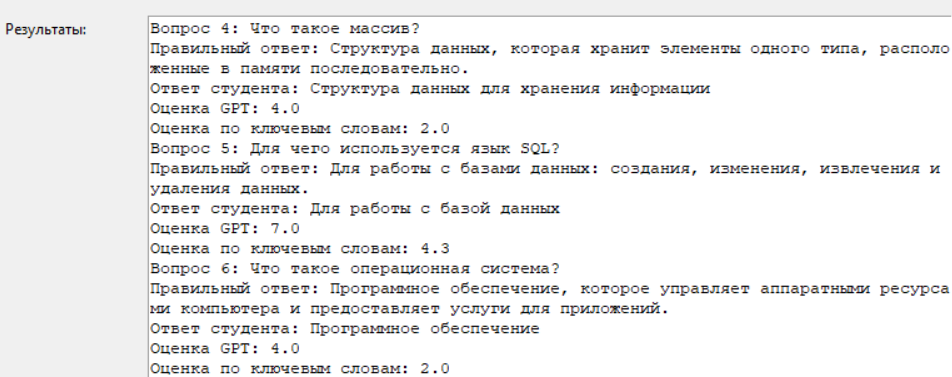


Рисунок 4 – Проверка на неполных ответах

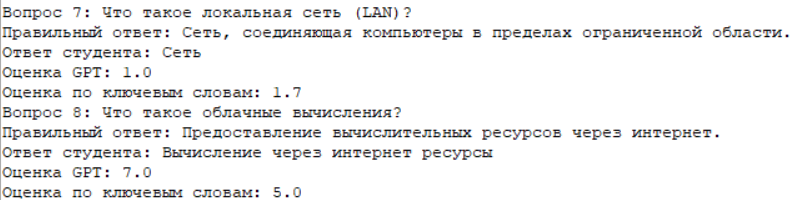


Рисунок 5 – Проверка на неполных ответах

Выводятся проведённые оценки с помощью Chat GPT и c помощью ключевых слов. При чём при повторном запуске программы и введении тех же самых ответов, оценки не будут отличаться от оценок после первого запуска программы.

# Заключение

В результате выполнения практической работы были исследованы методы анализа и обработки текстовой информации. После была создана программа, которая успешно автоматизирует проверку ответов студентов. Использование лемматизации и интеграции с Chat GPT позволяет получить точные и объективные оценки. Программа удобна в использовании и может быть применена в образовательных учреждениях для улучшения качества и скорости проверки знаний студентов.

# Литература

1. Документация по pymorphy2 [электронный ресурс]

https://pymorphy2.readthedocs.io/en/stable/user/guide.html

1. Сервис ProxyAPI [электронный ресурс]

https://console.proxyapi.ru/

1. Статьи на сайте «Habr» [электронный ресурс]

<https://habr.com/ru/>

1. Chat GPT [электронный ресурс]

https://openai.com/chatgpt/overview/

# Приложение 1

import tkinter as tk

from tkinter import messagebox, filedialog

import requests

import json

import string

import logging

import pymorphy3

# Настройка логирования

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Инициализация лемматизатора

morph = pymorphy3.MorphAnalyzer()

def lemmatize\_and\_clean\_text(text):

"""Лемматизация и очистка текста от пунктуации, предлогов и союзов."""

tokens = text.split()

tokens = [word.lower().strip(string.punctuation) for word in tokens]

lemmatized\_tokens = [

morph.parse(token)[0].normal\_form

for token in tokens

if token and morph.parse(token)[0].tag.POS not in {'PREP', 'CONJ'} # Исключаем предлоги и союзы

]

return lemmatized\_tokens

questions = []

current\_question\_index = 0

student\_answers = []

def load\_questions():

"""Функция для загрузки файла с вопросами."""

global questions, current\_question\_index, student\_answers

file\_path = filedialog.askopenfilename(

title="Открыть файл с вопросами",

filetypes=[("JSON файлы", "\*.json"), ("Все файлы", "\*.\*")]

)

if not file\_path:

return

try:

with open(file\_path, "r", encoding="utf-8") as file:

questions = json.load(file)

if not isinstance(questions, list) or not all("question" in q and "answer" in q for q in questions):

raise ValueError("Неверный формат файла. Ожидается список с объектами 'question' и 'answer'.")

current\_question\_index = 0

student\_answers = [{} for \_ in range(len(questions))]

display\_question()

except Exception as e:

logging.error(f"Не удалось загрузить файл: {e}")

messagebox.showerror("Ошибка", f"Не удалось загрузить файл: {e}")

def display\_question():

"""Отображает текущий вопрос на экране."""

global current\_question\_index, questions

if current\_question\_index < len(questions):

question\_label.config(text=f"Вопрос {current\_question\_index + 1}: {questions[current\_question\_index]['question']}")

student\_answer\_entry.delete("1.0", tk.END)

if student\_answers[current\_question\_index].get("student\_answer"):

student\_answer\_entry.insert("1.0", student\_answers[current\_question\_index]["student\_answer"])

else:

messagebox.showinfo("Завершено", "Все вопросы пройдены. Отображение результатов.")

display\_results()

def save\_current\_answer():

"""Сохраняет текущий ответ студента."""

global current\_question\_index, student\_answers

if 0 <= current\_question\_index < len(student\_answers):

student\_answer = student\_answer\_entry.get("1.0", tk.END).strip()

student\_answers[current\_question\_index]["student\_answer"] = student\_answer

else:

logging.error(f"Недопустимый индекс вопроса: {current\_question\_index}")

def check\_answer(index):

"""Проверяет ответ студента для указанного вопроса."""

global questions, student\_answers

question = questions[index]["question"]

correct\_answer = questions[index]["answer"]

student\_answer = student\_answers[index].get("student\_answer", "").strip()

if not student\_answer:

return None

try:

# Лемматизация ключевых слов из правильного ответа

correct\_answer\_tokens = set(lemmatize\_and\_clean\_text(correct\_answer))

student\_answer\_tokens = set(lemmatize\_and\_clean\_text(student\_answer))

# Подсчет совпадающих ключевых слов

matched\_keywords = len(correct\_answer\_tokens & student\_answer\_tokens)

keyword\_score = round((matched\_keywords / len(correct\_answer\_tokens)) \* 10, 1) if correct\_answer\_tokens else 0

# GPT-оценка

prompt = (

f"Вопрос: {question}\n"

f"Правильный ответ: {correct\_answer}\n"

f"Ответ студента: {student\_answer}\n"

"Оцени ответ студента по 10-ти бальной системе, где 0 - абсолютно неверно, 10 - идеальный ответ. Напечатай только число."

)

url = "https://api.proxyapi.ru/openai/v1/chat/completions"

headers = {

"Authorization": f"Bearer sk-V5Nsf1jfsU2JMuUNtACHMJ3LRsl9b5Pt",

"Content-Type": "application/json"

}

data = {

"model": "gpt-4",

"messages": [

{"role": "system", "content": "Ты являешься ассистентом преподавателя, который проверяет ответы студентов."},

{"role": "user", "content": prompt}

]

}

response = requests.post(url, headers=headers, json=data, timeout=30)

response.raise\_for\_status()

result = response.json()

gpt\_feedback = float(result["choices"][0]["message"]["content"].strip())

# Комбинированная оценка

combined\_score = round((gpt\_feedback + keyword\_score) / 2, 1)

student\_answers[index].update({

"question": question,

"correct\_answer": correct\_answer,

"student\_answer": student\_answer,

"gpt\_score": gpt\_feedback,

"keyword\_score": keyword\_score,

"combined\_score": combined\_score

})

except requests.RequestException as e:

logging.error(f"Ошибка соединения с API: {e}")

messagebox.showerror("Ошибка", f"Ошибка соединения с API: {e}")

except Exception as e:

logging.error(f"Произошла ошибка при проверке ответа: {e}")

messagebox.showerror("Ошибка", f"Произошла ошибка при проверке ответа: {e}")

def check\_all\_answers():

"""Проверяет ответы для всех вопросов."""

for i in range(len(questions)):

check\_answer(i)

display\_results()

def next\_question():

"""Переход к следующему вопросу."""

global current\_question\_index

save\_current\_answer()

if current\_question\_index < len(questions) - 1:

current\_question\_index += 1

display\_question()

else:

messagebox.showinfo("Конец", "Вы достигли конца вопросов.")

def previous\_question():

"""Переход к предыдущему вопросу."""

global current\_question\_index

save\_current\_answer()

if current\_question\_index > 0:

current\_question\_index -= 1

display\_question()

def display\_results():

"""Отображает результаты всех ответов."""

results\_text.delete("1.0", tk.END)

for i, ans in enumerate(student\_answers, start=1):

if "question" in ans:

results\_text.insert(tk.END, f"Вопрос {i}: {ans['question']}\n")

results\_text.insert(tk.END, f"Правильный ответ: {ans['correct\_answer']}\n")

results\_text.insert(tk.END, f"Ответ студента: {ans['student\_answer']}\n")

results\_text.insert(tk.END, f"Оценка GPT: {ans['gpt\_score']}\n")

results\_text.insert(tk.END, f"Оценка по ключевым словам: {ans['keyword\_score']}\n")

results\_text.insert(tk.END, f"Комбинированная оценка: {ans['combined\_score']}\n\n")

root = tk.Tk()

root.title("Проверка ответов студентов")

question\_label = tk.Label(root, text="Загрузите файл с вопросами.", wraplength=500, justify="left")

question\_label.grid(row=0, column=0, columnspan=3, sticky="w", padx=10, pady=5)

tk.Label(root, text="Ответ студента:").grid(row=1, column=0, sticky="w", padx=10, pady=5)

student\_answer\_entry = tk.Text(root, height=5, width=60)

student\_answer\_entry.grid(row=1, column=1, columnspan=2, padx=10, pady=5)

load\_button = tk.Button(root, text="Загрузить файл", command=load\_questions)

load\_button.grid(row=2, column=0, sticky="e", padx=10, pady=10)

prev\_button = tk.Button(root, text="Назад", command=previous\_question)

prev\_button.grid(row=2, column=1, sticky="e", padx=10, pady=10)

next\_button = tk.Button(root, text="Далее", command=next\_question)

next\_button.grid(row=2, column=2, sticky="w", padx=10, pady=10)

check\_all\_button = tk.Button(root, text="Проверить все ответы", command=check\_all\_answers)

check\_all\_button.grid(row=3, column=1, sticky="w", padx=10, pady=10)

results\_label = tk.Label(root, text="Результаты:")

results\_label.grid(row=4, column=0, sticky="nw", padx=10, pady=5)

results\_text = tk.Text(root, height=20, width=80, state="normal")

results\_text.grid(row=4, column=1, columnspan=2, padx=10, pady=5)

root.mainloop()

# Приложение 2

[

{

"question": "Что такое ООП?",

"answer": "Парадигма программирования, основанная на объектах."

}

{

"question": "Что такое алгоритм?",

"answer": "Последовательность шагов, предназначенных для решения задачи."

},

{

"question": "Какие бывают основные типы данных в программировании?",

"answer": "Числовые, строковые, логические, массивы, списки и словари."

},

{

"question": "Что такое массив?",

"answer": "Структура данных, которая хранит элементы одного типа, расположенные в памяти последовательно."

},

{

"question": "Для чего используется язык SQL?",

"answer": "Для работы с базами данных: создания, изменения, извлечения и удаления данных."

},

{

"question": "Что такое операционная система?",

"answer": "Программное обеспечение, которое управляет аппаратными ресурсами компьютера и предоставляет услуги для приложений."

},

{

"question": "Что такое локальная сеть (LAN)?",

"answer": "Сеть, соединяющая компьютеры в пределах ограниченной области."

},

{

"question": "Что такое облачные вычисления?",

"answer": "Предоставление вычислительных ресурсов через интернет."

}

]