|  |
| --- |
| Оглавление  [1. Структура массива псевдоокончаний, используемого для поиска сказуемых в вопросе. 2](#_Toc185782066)  [2. Содержание массива «черный список» слов в вопросе, не являющихся сказуемыми. 2](#_Toc185782067)  [3. Структура функции поиска номера слова в вопросе с псевдоокончанием сказуемого. 2](#_Toc185782068)  [4. Структура функций для преобразования первых букв в тексте вопроса и ответа 3](#_Toc185782069)  [5. Процедура учета знаков препинания в предложении вопроса. 3](#_Toc185782070)  [6. Процедура разбиения текста вопроса на отдельные слова методом split(). 3](#_Toc185782071)  [7. Процедура поиска сказуемого в вопросе и формирования для него регулярного выражения 4](#_Toc185782072)  [8. Процедура формирования регулярного выражения для подлежащего в вопросе методами slice() и join() 4](#_Toc185782073)  [9. Процедура формирования предложения для ответа на вопрос из базы знаний методом test() 5](#_Toc185782074)  [10. Процедура и код формирования ответа на вопрос с синомичными сказуемыми 5](#_Toc185782075)  [11. Возможности библиотеки jQuery для организации мультимедийного диалога с Базой знаний симулятора установки на Web-странице. 6](#_Toc185782076)  [12. Общая структура сайта и разметки диалоговой Web-страницы для создания мультимедийного диалога с Базой знаний симулятора установки. 6](#_Toc185782077)  [13. Программные модули, добавляемые в файл генератора ответов dialog.js для функционирования мультимедийного диалога с Базой знаний на Web-странице. 7](#_Toc185782078)  [14. Назначение и общая структура функции dialog\_window() для создания диалогового окна 7](#_Toc185782079)  [15. Структура и стиль оформления общего блока div для диалогового окна 7](#_Toc185782080)  [16. Структура и стиль оформления блока div для полосы с кнопкой «Нажми, чтобы спросить» для запуска функции выдвижения диалогового окна на центр страницы 9](#_Toc185782081)  [17. Структура функции openDialog() плавного выдвижения-закрытия диалогового окна поверх Web-страницы 10](#_Toc185782082)  [18. Структура и стиль оформления блока div для выдачи ответа на вопрос в окно истории ответов 10](#_Toc185782083)  [19. Структура и стиль оформления блока div для поля формы input и кнопкой «Спросить» для запуска функции получения ответа из Базы знаний и выдачи его в блок ответа диалогового окна 10](#_Toc185782084)  [20. Общая структура функция ask() обработки вопроса и выдачи ответа в диалоговое окно 11](#_Toc185782085) |

# Структура массива псевдоокончаний, используемого для поиска сказуемых в вопросе.

В русском языке окончания сказуемых часто зависят от спряжения, возвратности, рода и числа. Массив псевдоокончаний endings[] помогает обрабатывать эти особенности через регулярные выражения.

|  |
| --- |
| var endings = [  ["ет", "(ет|ут|ют)"], // Окончание для 3-го лица ед. числа глаголов I спряжения  ["ут", "(ет|ут|ют)"], // Окончание для 3-го лица мн. числа глаголов I спряжения  ["ится", "(ит|ат|ят)ся"], // Возвратное окончание II спряжения  ["ано", "ано"], // Краткое прилагательное среднего рода  ["такое", "– это"] // Исключение для "что такое ...?"  ]; |

Первая колонка: фактическое окончание, встречающееся в вопросе.

Вторая колонка: регулярное выражение, которое используется для проверки совпадений.

Для конструкций типа «что такое ...?», где используется "псевдосказуемое" (например, такое), добавляется строка:

|  |
| --- |
| ["такое", "– это"] |

# Содержание массива «черный список» слов в вопросе, не являющихся сказуемыми.

Массив «чёрный список» (blacklist[]) служит для исключения из обработки слов, которые имеют окончания, схожие со сказуемыми, но на самом деле сказуемыми не являются. Это помогает избежать ложного срабатывания при поиске сказуемых в вопросах и записи базы знаний.

|  |
| --- |
| var blacklist = ["замена", "замены", "атрибут", "маршрут", "член", "нет"]; |

Во время обработки текста каждое слово проверяется не только на соответствие псевдоокончаниям, но и на присутствие в массиве blacklist[]. Если слово находится в этом списке, оно пропускается, даже если формально его окончание подходит под шаблон сказуемого.

# Структура функции поиска номера слова в вопросе с псевдоокончанием сказуемого.

Функция getEnding(word) проверяет, является ли слово сказуемым:

|  |
| --- |
| function getEnding(word) {  if (blacklist.indexOf(word) !== -1) return -1; // исключение из обработки  for (var j = 0; j < endings.length; j++) {  if (word.endsWith(endings[j][0])) return j; // возвращает индекс окончания  }  return -1; // не найдено  } |

Для анализа сказуемого на совпадения его окончания с соответствующим ему псевдоокончанием в массиве псевдоокончаний endings[] определяется вспомогательная функция getEnding(word), в которой вначале сразу же проверяется, не находится ли рассматриваемое слово word в списке исключений массива blacklist[] и поэтому его нужно исключить из дальнейшего рассмотрения

Затем в цикле для всех записей в первом столбце массива endings[]производится проверка, не имеет ли это слово одно из псевдоокончаний, характерных для сказуемого

# Структура функций для преобразования первых букв в тексте вопроса и ответа

Функция small()для преобразования первой буквы в тексте ответа в прописную:

|  |
| --- |
| function small(str) {  return str[0].toLowerCase() + str.slice(1);  } |

Функция big()для преобразования первой буквы в тексте ответа в заглавную:

|  |
| --- |
| function big(str) {  return str[0].toUpperCase() + str.slice(1);  } |

# Процедура учета знаков препинания в предложении вопроса.

Знаки препинания отделяются пробелами.

Создается массив separators, содержащий все символы пунктуации, которые нужно обработать. Каждому символу из separators присваивается пробел перед ним.

Это выполняется для того, чтобы знаки препинания, прикрепленные к словам (например, "металл," или "что?"), не сливались со словами и корректно разделялись при дальнейшей обработке:

|  |
| --- |
| var separators = "'\",.!?()[]\\/";  for (var i = 0; i < separators.length; i++) {  txt = txt.replace(separators[i], " " + separators[i]);  } |

# Процедура разбиения текста вопроса на отдельные слова методом split().

После учета знаков препинания текст вопроса хранится в переменной txt. Например(Придумай свой текст) (Ну и пробел перед ‘?’ нужен)):

|  |
| --- |
| var txt = "Что такое металл ?"; |

Метод split() применяется с разделителем ' ' (пробел). Это позволяет разделить текст на массив слов, где каждое слово или знак препинания становится отдельным элементом массива:

|  |
| --- |
| var words = txt.split(' '); |

Как итог, в массив words добавляются все элементы строки, разделенные пробелами, включая знаки препинания.

Для примера текста "Что такое металл ?" массив будет выглядеть так:

|  |
| --- |
| words = ["Что", "такое", "металл", "?"]; |

# Процедура поиска сказуемого в вопросе и формирования для него регулярного выражения

Цель этой процедуры — найти сказуемое в вопросе и преобразовать его в шаблон регулярного выражения, чтобы сопоставлять с данными из базы знаний.

1. **Перебор слов в вопросе:**

После разбиения текста на массив слов (words) начинается цикл по каждому слову:

|  |
| --- |
| for (var i = 0; i < words.length; i++) {  var ending = getEnding(words[i]); // Проверка слова на псевдоокончание  } |

1. **Проверка псевдоокончаний:**

Функция getEnding(word) проверяет, соответствует ли слово сказуемому, используя массив псевдоокончаний (endings). Если найдено соответствие, возвращается индекс псевдоокончания.

1. **Создание шаблона для сказуемого**:

Если getEnding() возвращает индекс (то есть слово распознано как сказуемое), это слово преобразуется: у псевдоокончания в слове отсекается его часть. И добавляется шаблон из второго столбца массива endings[]

|  |
| --- |
| if (ending >= 0) {  words[i] = words[i].slice(0, -endings[ending][0].length) + endings[ending][1];  var predicate = new RegExp(words[i]); // Формируется регулярное выражение  } |

1. **Дополнительная обработка кратких прилагательных**:

Если сказуемое является кратким прилагательным (например, "явен", "заметен"), то для шаблона также захватывается следующее слово, так как прилагательное часто употребляется в связке с существительным:

|  |
| --- |
| if (endings[ending][0] == endings[ending][1]) {  predicate = new RegExp(words[i] + " " + words[i + 1]);  i++; // Пропускаем следующее слово  } |

1. **Сохранение шаблона**:

После создания регулярное выражение для сказуемого сохраняется в переменной predicate и будет использоваться для поиска в базе знаний.

# Процедура формирования регулярного выражения для подлежащего в вопросе методами slice() и join()

1. **Выделение слов, относящихся к подлежащему**:

После нахождения сказуемого в вопросе (на этапе поиска), подлежащее — это все слова, идущие за сказуемым. Эти слова выделяются с помощью метода slice() для массива слов words:

|  |
| --- |
| var subject\_words = words.slice(i + 1); |

1. **Объединение слов в строку с использованием шаблона**:

Для формирования регулярного выражения пробелы между словами заменяются на шаблон .\*, который означает произвольную последовательность символов (любая строка):

|  |
| --- |
| var subject\_string = subject\_words.join(".\*"); |

1. **Добавление шаблонов в начале и конце строки**:

К строке добавляются .\* в начале и в конце, чтобы учитывать случаи, когда подлежащее может быть частью более длинного текста:

|  |
| --- |
| var subject = new RegExp(".\*" + subject\_string + ".\*"); |

1. **Проверка длины строки подлежащего**:

Регулярное выражение создается только если длина подлежащего превышает 3 символа, чтобы избежать ложных совпадений с короткими словами:

|  |
| --- |
| if (subject\_string.length > 3) {  var subject = new RegExp(".\*" + subject\_string + ".\*");  } |

# Процедура формирования предложения для ответа на вопрос из базы знаний методом test()

Ячейки второго столбца проверяются на соответствие регулярному выражению predicate, полученному из сказуемого.

Регулярное выражение subject, полученное для подлежащего, используется для проверки на соответсвие ячеек как первого, так и третьего столбцов в двумерном массиве базы знаний, поскольку одинаковые по смыслу предложения в исходном тексте могут быть сформированы с переставленными в предложении субъектной и объектными частями.

|  |
| --- |
| //поиск совпадений с шаблонами среди связей семантической сети  for (var j = 0; j < knowledge.length; j++)  {  //проверяемое условие на совпадение регулярных выражений для  //сказуемого и подлежащего в вопросе с записями в массиве Базы  //знаний  if (predicate.test(knowledge[j][1]) &&  (subject.test(knowledge[j][0]) ||  subject.test(knowledge[j][2])))  //создание предложения для ответа из найденной по заданному  //вопросу строки из массива Базы знаний  {  answer+=big(knowledge[j][0] + " " + knowledge[j][1] + " " + knowledge[j][2] + ". ");  result = true;  }  } |

# Процедура и код формирования ответа на вопрос с синомичными сказуемыми

Поскольку в вопросе может быть использовано сказуемое, синонимичное (совпадающее по смыслу, но не совдадающее по написанию) сказуемому использованного в тексте, предусмотрен повторный проход по строкам базы знаний без проверки сказуемого.

|  |
| --- |
| //если совпадений с двумя шаблонами нет  if (!result) {  for (var j = 0; j < knowledge.length; j++) {  if (subject.test(knowledge[j][0]) || subject.test(knowledge[j][2])) {  //создание простого предложения из найденной строки в массиве Базы  //знаний  answer += big(knowledge[j][0] + " " + knowledge[j][1] + " " + knowledge[j][2] + ". ");  }  }  } |

# Возможности библиотеки jQuery для организации мультимедийного диалога с Базой знаний симулятора установки на Web-странице.

* Может обращаться к любому элементу объектной модели Web-документа DOM и предлагает механизм селекторов
* Управление событиями
* Осуществляет различные визуальные эффекты с изменением внешнего вида страницы с использованием CSS
* Имеет различные JavaScript-плагины, предназначенные для создания элементов пользовательских интерфейсов, например, диалоговых окон

# Общая структура сайта и разметки диалоговой Web-страницы для создания мультимедийного диалога с Базой знаний симулятора установки.

1. **В папку сайта добавить файлы:**

* библиотеку jQuery – файл jquery.js,
* файл разработанного в ЛР по Теме 12 программного кода на языке JavaScript для обработки диалога, например, с именем dialog.js с функцией формирования диалогового окна, например, с именем dialog\_window();
* файл внешней таблицы разработанных для диалога CSS-стилей, сохраненного, например, с именем style.css.

1. **В HTML-разметку диалоговой Web-страницы добавить следующие элементы:**

* строку с тегом <script> для подключения файла библиотеки jQuery jquery.js;
* строку с тегом <script> для подключения аудио сервиса Yandex;
* строку с тегом <script> для подключения внешнего файла с программным кодом JavaScript dialog.js, в который добавить код, обеспечивающий интерфейс на основе диалогового окна;
* строку с тегом <link> для подключения внешней таблицы CSS-стилей из файла style.css;
* в теге <body> загрузить оператором onLoad() созданную дополнительно функцию организации интерфейса для диалогового окна для Web-страницы, например, с именем dialog\_window().

|  |
| --- |
| <html lang="ru" xmlns="http://www.w3.org/1999/xhtml">  <head>  <title>Диалог с базой знаний</title>  <meta charset="utf-8" />  <script src="jquery.js"></script>  <script src="dialog.js"></script>  <link href="style.css" type="text/css" rel="stylesheet"/>  </head>  <body onLoad="dialog\_window()">  …  </body>  </html> |

# Программные модули, добавляемые в файл генератора ответов dialog.js для функционирования мультимедийного диалога с Базой знаний на Web-странице.

**1. Флаговая переменная dialogOn** для отслеживания активности диалогового окна.

**2. Функции:**

* dialog\_window() – создание диалогового окна.
* openDialog() – плавное открытие/закрытие окна.
* ask() – обработка вопроса и выдача ответа​.

# Назначение и общая структура функции dialog\_window() для создания диалогового окна

Функция предназначена для создания диалогового окна с разделами для текста, истории, и формы ввода

Код добавляет HTML-разметку окна и предполагает добавление стилей через CSS:

|  |
| --- |
| <div id="dialog" class="dialog">  <!-- Полоса с кнопкой "Нажми, чтобы спросить" -->  <div class="label" onclick="openDialog()">Нажми, чтобы спросить!</div>  <!-- Заголовок "История" -->  <div class="header">История:</div>  <!-- Область для отображения истории вопросов и ответов -->  <div class="history" id="history"></div>  <!-- Поле ввода вопроса и кнопка "Спросить" -->  <div class="question">  <input id="Qdialog" placeholder="Введите вопрос" />  <br />  <button onclick="ask('Qdialog')">Спросить</button>  </div>  </div> |

# Структура и стиль оформления общего блока div для диалогового окна

Структура, как я пон, – из прошлого задания. Хз, что именно хочет дед.

Структура:

|  |
| --- |
| <div id="dialog" class="dialog">  <!-- Полоса с кнопкой "Нажми, чтобы спросить" -->  <div class="label" onclick="openDialog()">Нажми, чтобы спросить!</div>  <!-- Заголовок "История" -->  <div class="header">История:</div>  <!-- Область для отображения истории вопросов и ответов -->  <div class="history" id="history"></div>  <!-- Поле ввода вопроса и кнопка "Спросить" -->  <div class="question">  <input id="Qdialog" placeholder="Введите вопрос" />  <br />  <button onclick="ask('Qdialog')">Спросить</button>  </div>  </div> |

Стиль (что он вообще хочет…):

|  |
| --- |
| .dialog{  position:fixed;  top:10%;  left:100%;  width:800px;  height:800px;  margin-left:-30px;  margin-top:-25px;  padding:10px;  border:#000 2px solid;  border-radius:10px;  background:#fff;  color:#f00;  z-index:1500;  font-size:12pt;  } |

Это тоже стили, но уже не для “общего блока div”, а для вложенных div’ов, СКОРЕЕ ВСЕГО НЕ НАДО:

|  |
| --- |
| .dialog .header{  text-align:center;  margin-top:-430px;  font-size:20px;  } |

|  |
| --- |
| .dialog .history{  height: 730px;  overflow-x:hidden;  overflow-y:scroll;  } |

|  |
| --- |
| .dialog .history .question{  margin-left:40px;  background:#99f;  margin:5px;  width:100%;  border-radius: 5px;  padding:5px;  }  .dialog button{  margin:2px 20px;  width:95%;  border:1px #666 solid;  border-radius: 5px;  font-size:20px;  }  input{  font-size:20px;  width:100%;  border-radius:10px;  padding:2px 10px;  } |

P.S. Крч кринж какой-то

# Структура и стиль оформления блока div для полосы с кнопкой «Нажми, чтобы спросить» для запуска функции выдвижения диалогового окна на центр страницы

Структура блока div для полосы с кнопкой “Нажми, чтобы спросить”

|  |
| --- |
| <div class='label' onclick='openDialog()'>  Нажми, чтобы спросить!  </div> |

Стиль:

|  |
| --- |
| .dialog .label {  transform: rotate(270deg);  width: 430px;  text-align: center;  color:#f00;  overflow:hidden;  display:inline-block;  margin-left:-26%;  margin-top:50%;  cursor: pointer;  } |

# Структура функции openDialog() плавного выдвижения-закрытия диалогового окна поверх Web-страницы

|  |
| --- |
| function openDialog() {  if(dialogOn) {  $("#dialog").animate({"margin-left":"-30px"}, 1000);  dialogOn = false;  } else {  $("#dialog").animate({"margin-left":"-1250px"}, 1000);  dialogOn = true;  }  } |

# Структура и стиль оформления блока div для выдачи ответа на вопрос в окно истории ответов

Блок div для выдачи ответа на вопрос:

|  |
| --- |
| <div class=’history’ id=’history’>  </div> |

Стиль:

|  |
| --- |
| .dialog .history {  height: 730px;  overflow-y: scroll;  } |

# Структура и стиль оформления блока div для поля формы input и кнопкой «Спросить» для запуска функции получения ответа из Базы знаний и выдачи его в блок ответа диалогового окна

Блок div для поля формы формы input и кнопкой «Спросить»:

|  |
| --- |
| <div class='question'>  <input id='Qdialog' placeholder='Введите вопрос'/><br>  <button onclick='ask(\"Qdialog\")'>  Спросить  </button>  </div> |

Стиль

|  |
| --- |
| .dialog input, .dialog button, .dialog .history{  margin:2px 25px;  width:85%;  border:1px #666 solid;  border-radius: 5px;  font-size:20px;  }  .dialog .history .question {  margin:5px;  width:100%;  border-radius: 5px;  padding:5px;  margin-left:40px;  background:#99f;  } |

# Общая структура функция ask() обработки вопроса и выдачи ответа в диалоговое окно

|  |
| --- |
| function ask(questionInput) {  var question = document.getElementById(questionInput).value;  var newDiv = document.createElement("div");  newDiv.className = 'question';  newDiv.innerHTML = question;  document.getElementById("history").appendChild(newDiv);  newDiv = document.createElement("div");  newDiv.className = 'answer';  newDiv.innerHTML = getAnswer(question);  document.getElementById("history").appendChild(newDiv);  } |

**ВОПРОСЫ КАКОЙ-ТО КР ПО БАЗЕ ЗНАНИЙ И ДИАЛОГУ**

### 1. Семантика и знание

\*\*Семантика\*\* — это раздел лингвистики, изучающий смысл слов и предложений. Она охватывает, как смысл формируется и передается в языке.

\*\*Знание\*\* — это информация, понятия и навыки, которые человек или система приобретают через опыт или обучение. В контексте искусственного интеллекта знание может быть формализовано для обработки и анализа.

### 2. Искусственный интеллект

\*\*Искусственный интеллект (ИИ)\*\* — это область информатики, занимающаяся созданием систем, способных выполнять задачи, требующие интеллекта, такие как обучение, понимание естественного языка и решение проблем. Основная задача ИИ — разработка алгоритмов и моделей, которые могут имитировать человеческое мышление и поведение.

### 3. Семантическая сеть

\*\*Семантическая сеть\*\* — это структура, представляющая знания в виде узлов (понятий) и рёбер (отношений) между ними. В текстовой информации семантическая сеть может быть использована для организации информации, где каждое понятие связано с другими понятиями, что упрощает поиск и обработку данных.

### 4. Членение предложений на семантические блоки

\*\*Триада\*\* — это структура, состоящая из трех элементов: субъект, предикат и объект. Принцип членения предложений в тексте на семантические блоки позволяет выделить основные смысловые единицы, что облегчает анализ и обработку текста.

### 5. Представление Базы знаний

База знаний может быть представлена в виде структурированных данных, таких как таблицы, графы или семантические сети, где хранятся факты, правила и взаимосвязи между ними.

### 6. Хранение семантической сети как списка дуг

Хранение семантической сети как списка дуг (триад) предполагает использование коллекции связей между узлами, где каждая дуга представляет отношение между двумя понятиями. Это может быть реализовано в виде массива или списка, где каждая запись содержит три элемента: субъект, предикат и объект.

### 7. Обработка сложных предложений

Обработка сложных предложений для представления их в триады требует анализа синтаксической структуры предложения. Это может включать разбиение на составные части, идентификацию главного сказуемого и определение подлежащего.

### 8. Хранение Базы знаний в виде текстового двумерного массива

Хранение Базы знаний в виде текстового двумерного массива предполагает, что данные организованы в таблицы, где строки представляют факты или правила, а столбцы — различные атрибуты этих фактов.

### 9. Хранение графических и мультимедийных объектов

Графические и мультимедийные объекты могут быть встроены в текст с помощью ссылок или встроенного кода. Это позволяет отображать изображения, видео и анимации в контексте текстовой информации.

### 10. Структура диалогового вопроса

Структура диалогового вопроса для получения точечного ответа может включать ключевые слова, такие как подлежащее, сказуемое и объект, а также дополнительные элементы, уточняющие запрос, например, время или место.

### 11. Методика отбора ответа

Методика отбора ответа из семантической сети может основываться на использовании ключевых слов из вопроса для поиска соответствующих узлов и рёбер, а также применения алгоритмов, таких как поиск по графу.

### 12. Проблемы при поиске сказуемых

Проблемы при поиске сказуемых могут включать неоднозначность в языке, сложные конструкции предложений, а также отсутствие четких маркеров, указывающих на сказуемое.

### 13. Регулярное выражение

\*\*Регулярное выражение\*\* — это последовательность символов, определяющая шаблон для поиска в текстах. Они используются для поиска, замены и валидации текстовых данных.

### 14. Назначение регулярных выражений

Регулярные выражения для сказуемого и подлежащего в вопросе помогают выявлять и выделять ключевые элементы предложения, что упрощает анализ структуры вопроса.

### 15. Этапы процедуры поиска ответа

Этапы процедуры поиска точечного ответа могут включать:

1. Анализ вопроса.

2. Определение ключевых компонентов (подлежащее, сказуемое).

3. Поиск соответствий в Базе знаний.

4. Формирование ответа.

### 16. Алгоритм процедуры поиска

Алгоритм может включать:

1. Разделение вопроса на составляющие.

2. Идентификацию сказуемого и подлежащего.

3. Поиск по семантической сети.

4. Формирование и возврат ответа.

### 17. Классы, функции и методы JavaScript

Классы и функции JavaScript для организации диалога могут включать обработчики событий, методы для работы с DOM, а также функции для анализа текста и взаимодействия с Базой знаний.

### 18. Структура таблицы псевдоокончаний

Структура таблицы псевдоокончаний для сказуемых может включать разные формы слов, их окончания и соответствующие правила, позволяющие правильно определять сказуемые в предложении.

### 19. Содержание массива псевдоокончаний

Содержание массива псевдоокончаний может включать окончания глаголов в различных формах, что позволяет алгоритму корректно определять сказуемые в зависимости от контекста.

### 20. Черный список подлежащих

Содержание массива «черный список» может включать слова, которые не могут быть подлежащими в контексте заданного предложения, например, предлоги или союзные слова, что помогает избежать ошибок в анализе.

Если у вас есть дополнительные вопросы или требуется более глубокое объяснение по какому-либо пункту, дайте знать!