Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «Программные средства кибертехнических систем»

на тему «Разработка экспертной системы»

Выполнили:

студенты группы 22ВВП1

Демин М. С.

Беляев Д. И.

Захаров А. С.

Приняли:

Дубинин В.Н.

Карамышева Н.С.

Пенза 2025

**Название**

Разработка экспертной системы

**Цель работы**

Изучение способов разработки экспертных систем и их реализации.

**Задание**  
Необходимо разработать экспертную систему для выбора оптимального языка программирования в зависимости от поставленной задачи.  
ЭС «помогает» подобрать язык, основываясь на предпочтениях пользователя, характере решаемых задач и ключевых требованиях к программному обеспечению.

**Ход работы:**

Выбор языка программирования будет зависеть от трех основных характеристик – **производительности**, **простоты разработки** и **гибкости и применимости**. Рассмотрим каждую из них подробнее.

### ****1. Производительность****

Под производительностью подразумевается эффективность работы языка в плане скорости выполнения кода, оптимизации использования памяти и поддержки многопоточности. Данный критерий особенно важен для разработки высоконагруженных систем, обработки больших объемов данных, работы с встраиваемыми системами и приложений, критичных к быстродействию (например, игры, финансовые системы, научные вычисления). Важно для тех, кто занимается разработкой высокоэффективного программного обеспечения и нуждается в максимальной оптимизации вычислений (табл. 3.1).

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Пояснение** | **От чего зависит** |
| **Скорость выполнения** | Время, затрачиваемое на выполнение кода | Тип языка (компилируемый/интерпретируемый), оптимизации компилятора |
| **Потребление памяти** | Объем памяти, требуемый для выполнения программы | Структуры данных, механизм сборки мусора |
| **Многопоточная обработка** | Возможность эффективного выполнения нескольких потоков | Поддержка многопоточности и асинхронности |
| **Время компиляции** | Скорость преобразования исходного кода в исполняемый | Оптимизация компилятора, сложность синтаксиса |
| **Производительность на больших данных** | Эффективность работы с объемными массивами информации | Оптимизация алгоритмов, работа с памятью |
| **Поддержка JIT-компиляции** | Возможность ускоренной работы через Just-in-Time компилятор | Наличие JIT (как в Java, Python, .NET) |
| **Эффективность работы с кэшем** | Оптимизация работы с кэш-памятью процессора | Архитектура языка, доступ к низкоуровневым API |

### ****2. Простота разработки****

Простота разработки включает в себя удобство написания и поддержки кода, уровень сложности синтаксиса, возможности отладки, а также наличие качественной документации. Этот критерий важен для быстрого освоения языка, повышения продуктивности работы разработчиков и снижения затрат на поддержку кода. Он особенно критичен для начинающих программистов, а также для тех, кто хочет минимизировать сложность работы с языком, не теряя при этом в функциональности (табл. 3.2).

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Пояснение** | **От чего зависит** |
| **Легкость освоения** | Простота синтаксиса и порог вхождения | Количество ключевых слов, логичность структуры |
| **Читаемость кода** | Удобство восприятия написанного кода | Ясность синтаксиса, необходимость явного объявления типов |
| **Наличие документации** | Качество и доступность обучающих материалов | Поддержка сообщества, официальные гайды |
| **Поддержка автоматического управления памятью** | Упрощает работу с памятью (нет необходимости освобождать вручную) | Наличие сборщика мусора (GC) |
| **Интерактивность разработки** | Возможность быстрого тестирования и отладки кода | Наличие REPL (Read-Eval-Print Loop), интерпретируемость |
| **Статическая vs. динамическая типизация** | Влияет на строгую проверку типов во время компиляции или выполнения | Строгость проверки типов, необходимость явного указания |
| **Наличие встроенных средств отладки** | Возможность легко находить ошибки | Нативные инструменты (debugger, профайлер) |
| **Простота настройки окружения** | Легкость установки и запуска кода | Инструменты сборки, зависимости |

**3. Гибкость и применимость**

Гибкость и применимость отражает универсальность языка, его пригодность для различных сфер программирования, поддержку разных парадигм и наличие развитой экосистемы инструментов и библиотек. Данный критерий важен для тех, кто работает в разных областях разработки (веб, мобильные приложения, искусственный интеллект, базы данных и т. д.) и хочет выбрать язык, который будет наиболее адаптивен под различные задачи (табл. 3.3).

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Пояснение** | **От чего зависит** |
| **Поддержка парадигм** | Возможность использовать разные стили программирования | Гибкость языка (ООП, функциональное, процедурное) |
| **Доступность библиотек** | Количество готовых решений для различных задач | Размер и активность сообщества, наличие стандартных библиотек |
| **Кроссплатформенность** | Возможность работы на разных ОС | Наличие виртуальной машины (JVM, .NET), компиляторы под разные платформы |
| **Широта применения** | Где можно использовать язык (веб, мобильная, встраиваемая разработка) | Поддержка инструментов, компиляторы |
| **Поддержка Web-разработки** | Возможность создавать серверные и клиентские веб-приложения | Наличие фреймворков (Django, Flask, Node.js) |
| **Поддержка мобильной разработки** | Используется ли для создания мобильных приложений | Наличие SDK (Kotlin, Swift, Flutter) |
| **Возможность работы с базами данных** | Легкость интеграции с СУБД | Наличие ORM, драйверов (SQLAlchemy, Hibernate) |
| **Безопасность** | Уровень защиты кода от уязвимостей | Механизмы защиты памяти, контроль доступа |
| **Инструменты тестирования** | Упрощает тестирование кода | Поддержка unit-тестирования, наличие встроенных фреймворков |

В этих таблицах собраны конфликтующие свойства, например, высокая производительность часто сопровождается сложностью разработки, а универсальность может снижать оптимизацию под конкретные задачи. Поэтому экспертная система должна определить, какие характеристики важны пользователю, и именно на их основе рекомендовать наиболее подходящий язык программирования.

### Общие требования к средствам реализации экспертной системы (ЭС)

1. **Архитектура и структура системы**
   * Модульная архитектура для гибкости и расширяемости.
   * Четкое разделение на интерфейсную часть (взаимодействие с пользователем), экспертную базу знаний (правила и данные) и движок вывода (логика принятия решений).
   * Возможность добавления новых критериев и языков программирования без значительных изменений в коде.
2. **База знаний**
   * Хранение информации о языках программирования, их характеристиках и применимости.
   * Представление знаний в виде логических правил, онтологий или базы данных.
   * Возможность обновления базы знаний без изменения основной логики работы системы.
3. **Механизм логического вывода**
   * Использование методов логического вывода (правдоподобного, дедуктивного, вероятностного).
   * Поддержка работы с нечёткой логикой для учёта субъективных предпочтений пользователя.
   * Возможность объяснения, почему был предложен тот или иной язык.
4. **Интерфейс взаимодействия с пользователем**
   * Удобный и интуитивно понятный интерфейс (графический или консольный).
   * Поддержка диалогового режима для уточнения предпочтений пользователя.
   * Визуализация критериев выбора (например, с использованием диаграмм или шкал приоритетов).
5. **Гибкость и настраиваемость**
   * Возможность выбора приоритетов (например, производительность важнее простоты разработки).
   * Опция расширения системы новыми параметрами и критериями оценки.
   * Поддержка различных типов пользователей (начинающий, опытный разработчик, архитектор ПО).
6. **Технологический стек**
   * Выбор языка программирования для реализации ЭС (например, Python с библиотеками для работы с экспертными системами – CLIPS, Pyke, scikit-learn, PyReasoner).
   * Возможность интеграции с внешними сервисами и базами знаний.
   * Кроссплатформенность для работы в вебе, на ПК или в облаке.
7. **Производительность и масштабируемость**
   * Оптимизация алгоритмов принятия решений для быстрого вывода результата.
   * Возможность работы с большим числом критериев без значительного падения скорости.
   * Использование кэширования для ускорения повторных запросов.
8. **Поддержка объяснимости решений**
   * Генерация отчётов или пояснений по выбору языка программирования.
   * История выбора пользователя с возможностью пересмотра критериев.
9. **Инструменты тестирования и валидации**
   * Наличие механизма тестирования корректности работы ЭС.
   * Валидация рекомендаций с помощью экспертных данных и реальных примеров.
10. **Безопасность**

* Защита от некорректного ввода данных пользователем.
* Ограничение прав доступа к базе знаний (если система используется в корпоративной среде).

Часть 2.

**База знаний экспертной системы по выбору языка программирования**

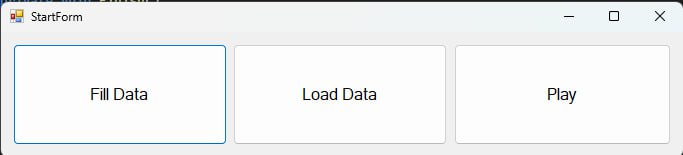
**Вопросы для пользователя**

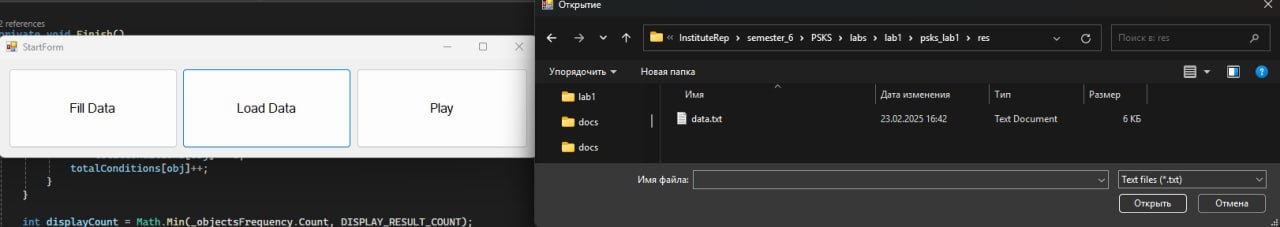
1. Вам важна высокая скорость выполнения кода?
2. Вам критично низкое потребление памяти?
3. Вы разрабатываете высоконагруженные системы?
4. Вам нужна поддержка многопоточной обработки?
5. Вам важна простота синтаксиса?
6. Вам важно наличие сборщика мусора?
7. Вам нужна статическая типизация?
8. Вам нужна кроссплатформенность?
9. Вы планируете использовать язык для веб-разработки?
10. Вы планируете использовать язык для мобильной разработки?
11. Вам нужна интеграция с базами данных?
12. Вам нужна поддержка ООП?
13. Вам нужна поддержка функционального программирования?
14. Вам важно наличие встроенных инструментов отладки?
15. Вы хотите, чтобы язык имел обширную документацию?
16. Вам важно наличие больших библиотек и фреймворков?
17. Вы хотите использовать язык для научных вычислений?
18. Вам нужна поддержка асинхронного программирования?
19. Вам важно, чтобы язык был удобен для написания скриптов?
20. Вы хотите, чтобы язык имел широкую поддержку сообщества?
21. Вам важна безопасность кода и защита от уязвимостей?
22. Вы планируете разрабатывать игры?
23. Вам нужна возможность писать код с минимальными зависимостями?
24. Вы хотите, чтобы язык был востребован на рынке труда?
25. Вам важна поддержка современных технологий (ML, AI, Big Data)?
26. Вы хотите язык с открытым исходным кодом?
27. Вам важно наличие мощного встроенного инструментария для тестирования?

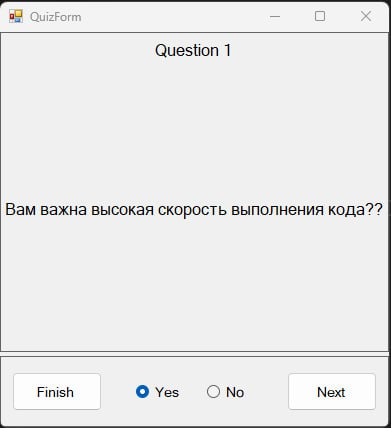
**Продукционные правила для выбора языка программирования**

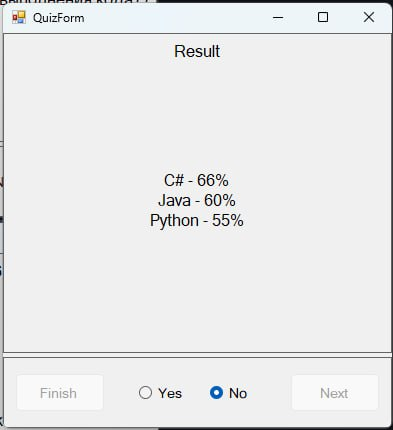
* C++:1,2,4,3,23,21
* Rust:1,21,4,23,2,8,25
* Python:5,16,17,6,18,19,25,20,26
* Java:8,12,21,20,4,16,11,24,18,14
* JavaScript:9,10,5,8,18,13,24,20
* Kotlin:10,12,18,24,13,8
* Swift:10,22,12,21,1,24
* Go:1,4,5,2,3,24,23
* Cs:21,27,8,12,24,11,4,13,14

**Разработка программы экспертной системы**

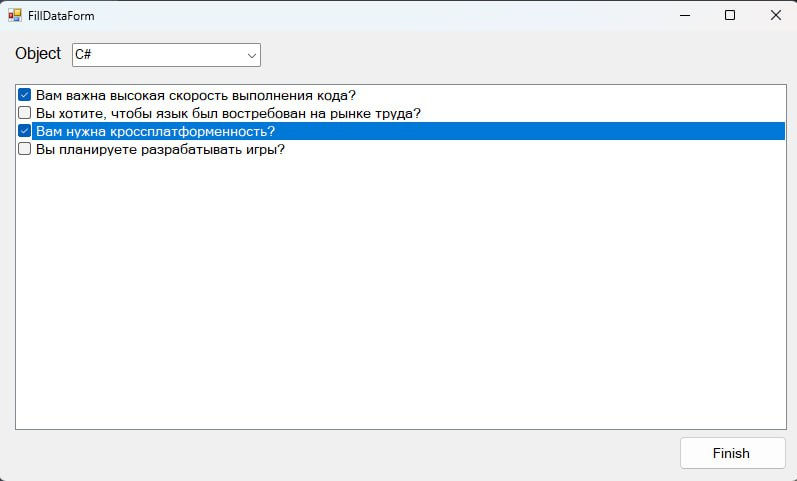
1. Начальный экран.  


2. Загрузка файла правил.  


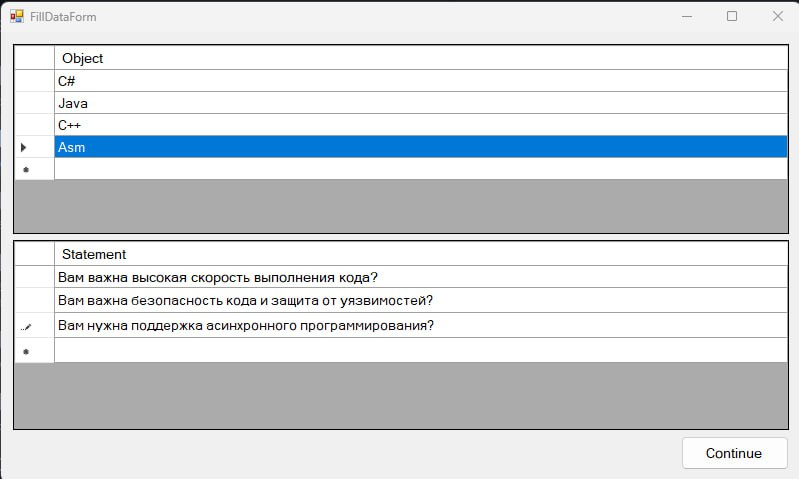
3. Форма ответов на вопросы.  


4. Форма результатов.  


5. Форма ручного ввода базы данных.



6. Форма просмотра базы данных.



**Вывод**

Спроектировали экспертную систему для предметной области связанной с языками программирования, которая ведет диалог с пользователем: задает вопросы, запоминает ответыпользователя.