**Лабораторная работа № 16**

Тема: Выбор и обоснование средств реализации проекта для разрабатываемой программной системы.

Цель работы:

1 изучить критерии выбора языка программирования. Научиться обосновывать выбор среды разработки в соответствии с критериями выбора языка программирования;

2 изучить основные парадигмы программирования;

3 изучить правила формирования листинга программы;

Задание 1 Критерии выбора языка программирования.

1. Изучить критерии выбора языка программирования. (если информация отсутствует в конспекте, то необходимо ее законспектировать).
2. Обосновать выбор языка программирования для решения поставленной задачи в соответствии с индивидуальным заданием.

Конструктор для создания сайта был выбран – Tilda.

Выбор данного конструктора обосновывается его простотой использования, Tilda предлагает интуитивно понятный интерфейс, который позволяет создать сайт даже без специальных знаний в области веб-разработки. Так же адаптивность дизайна. Сайты, созданные на Tilda, автоматически адаптируются под различные устройства (мобильные телефоны, планшеты, ПК). Tilda предоставляет мощные инструменты для работы с изображениями и видео.

1. HTML(HyperText Markup Language) — это язык разметки, который используется для создания структуры веб-страницы. Основные причины, почему HTML важен и полезен для создания сайтов:

* Структура контента: HTML определяет, как элементы (тексты, изображения, видео, ссылки) располагаются на странице. Он использует теги, которые помогают браузеру понять, какие части являются заголовками, параграфами, таблицами и т.д.
* SEO (поисковая оптимизация): HTML-теги, такие как <title>, <meta>, <h1>, <alt>, помогают поисковым системам понимать содержание веб-страницы и лучше индексировать её.
* Гибкость: HTML легко понимается и поддерживается всеми веб-браузерами, что делает его фундаментальным для создания любой веб-страницы.

1. CSS (Cascading Style Sheets) — Внешний вид и оформление

CSS отвечает за стиль и внешний вид веб-страницы. Он используется для задания цветов, шрифтов, размещения элементов и других визуальных аспектов. Почему CSS важен:

* Отделение стиля от структуры: CSS позволяет отделить внешний вид от содержания. HTML отвечает за структуру, а CSS — за оформление. Это упрощает как поддержку сайта, так и его масштабирование.
* Гибкость в дизайне: CSS даёт возможность создавать адаптивные и современные дизайны, которые хорошо работают как на мобильных устройствах, так и на больших экранах.
* Анимации и эффекты: CSS позволяет реализовывать анимации, переходы и сложные эффекты (например, анимацию при наведении курсора), что делает сайт более интерактивным и привлекательным для пользователей.
* Медиа-запросы: CSS помогает создавать адаптивные дизайны, которые подстраиваются под разные размеры экранов (например, мобильные устройства, планшеты).

1. JavaScript — Интерфейс и интерактивность

JavaScript — это язык программирования, который используется для создания интерактивных элементов на веб-сайте. Причины важности JavaScript:

* Интерактивность: JavaScript добавляет динамическое взаимодействие на веб-страницы. Например, это может быть работа с формами, валидация данных без перезагрузки страницы, создание всплывающих окон, анимация элементов.
* Динамическое изменение контента: JavaScript может изменять содержимое страницы на лету. Например, можно загружать новые данные с сервера, не перезагружая страницу (технология AJAX).
* Работа с API: JavaScript позволяет взаимодействовать с внешними сервисами и API, загружать и отправлять данные в реальном времени. Это основа для создания современных веб-приложений (например, чат-приложений, картографических сервисов).
* Кросс-платформенность: JavaScript поддерживается всеми современными браузерами, а благодаря технологиям вроде Node.js его можно использовать и на серверной стороне, что делает его универсальным языком.

Задание 2 Изучение парадигм программирования

Изучить различные парадигмы программирования. (если информация отсутствует в конспекте, то необходимо ее законспектировать). Провести сравнительную характеристику различных парадигм программирования. Результаты оформить в виде таблицы следующего вида:

Дайте характеристику следующим парадигмам программирования:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название парадигмы | Сущность парадигмы, основные идеи, принципы, объекты. | Языки  поддержив ающие  данную парадигму. (2-3 языка) | Достоинства | Недостатки | Примеры программ |
| Императивное программирование | Команда, состояние, контроль потока  Изменение состояния, Последовательность команд, Модульность, структурированность | С, Java, Python | Высокая производительность, контроль над памятью, множество библиотек и инструментов. | Низкий уровень абстракции, управление памятью требует дополнительных усилий, трудности в отладке. | Microsoft Word  Microsoft Excel  Adobe Photoshop  Apple Pages  Google Docs |
| Декларативное программирование | Акцент на достижении  Абстракция, читаемость, отделение логики от реализации,  Неопределенность реализации  Правила, Запросы,  Функции | SQL | Упрощение процесса разработки,  Высокая читаемость и поддерживаемость кода,  Возможность оптимизации выполнения,  Меньше ошибок благодаря абстракции. | Ограничение в контроле над процессом выполнения,  Могут возникнуть проблемы с производительностью в некоторых случаях,  Возможность затруднений в отладке из-за абстракции. | RxJS  Elm  Redux |
| Структурное программирование | Улучшение структуры программы, повышение ее читаемости и управляемости, а также упрощение процесса отладки.  Контрольный поток, декомпозиция, статическая проверка  Объектами являются функции и процедуры | C, Pascal | Упрощение понимания,  Упрощение отладки и тестирования  Модульность | Ограниченная гибкость,  Трудности с масштабируемостью,  Необходимость строгого планирования | Notepad  TextEdit  Pac-Man |
| Процедурное программирование | Фокусируется на последовательности инструкций  ДекомпозицияПовторяемость  Локальность  Структурированность  Код делится на процедуры и функции.  Процедуры используют параметры и возвращают результаты.  Управление потоком происходит с помощью последовательных, условных и циклических конструкций. | С, Pascal | Модульность  Структурированность | Сложность управления | Windows, macOS, Linux |
| Модульное программирование | Разделяет большие программы на модули,  Группировка взаимосвязанных функций  Инкапсуляция,  Абстракция,  Упрощение тестирования и отладки | Python, Java | Упрощение отладки,  Структурированность | Трудностям в понимании общего потока программы | Make  Maven  Django  Ruby on Rails |
| Объектно ориентированное программирование | ООП фокусируется на использовании объектов для моделирования реальных мировых сущностей и взаимодействия между ними.  Инкапсуляция  Наследование  Полиморфизм  Классы, Объекты | Python, Java, C# | Модульность  Внесение изменений и добавление функциональности | ООП может быть сложнее для понимания, особенно для новичков | Adobe Photoshop  AutoCAD  Android  iOS |
| Функциональное программирование | Рассматривает вычисление как выполнение математических функций и избегает изменения состояния и изменяемых данных.  Неизменемость данных |  |  |  | Haskell  Lisp  Scala |
| Логическое программирование | Фокусируется на описании фактов и правил  Факты,  Правила,  Запросы, |  | Возможность автоматического вывода новых фактов из заданных  Модульность | Программы на логическом языке могут становиться сложными и трудными для отладки | Пролог (использование правил и фактов для представления знаний)  Datalog |

3. Сделать выводы о том, какие из парадигм программирования используются наиболее часто.

Самой распространенной парадигмой является объектно-ориентированная парадигма, которая тесно связанная с императивным и функциональным программированием. Эти подходы активно интегрируются и используются в современном программировании для создания сложных и эффективных приложений.

4. Для своего индивидуального задания привести обоснование того какие парадигмы программирования вы будете использовать.

Императивная парадигма. Данная парадигма будет использована для создания основной логики взаимодействия на сайте. Она позволит детально описать пошаговую реализацию алгоритмов, связанных с функционалом платформы. Например, обработка пользовательских действий, таких как регистрация, авторизация, создание тем для обсуждения, и отправка сообщений. Используя императивный подход, можно последовательно описать, как программа должна реагировать на различные действия пользователя, от валидации данных до записи в базу данных и отправки уведомлений.

Объектно-ориентированная парадигма. Этот подход будет применен для моделирования сущностей сайта, таких как Пользователь, Сообщение, Тема обсуждения, Новость и т.д. Каждый из этих объектов будет иметь свои свойства (например, имя пользователя, текст сообщения, заголовок темы) и методы для взаимодействия с этими свойствами. Объектно-ориентированный подход поможет структурировать код, сделать его более читабельным и поддерживаемым, а также позволит эффективно управлять сложностью приложения, благодаря инкапсуляции, наследованию и полиморфизму.

Функциональная парадигма. Некоторые аспекты сайта, например, обработка данных или применение функций к коллекциям сообщений или тем, можно реализовать с помощью функциональной парадигмы. Используя чистые функции, можно обрабатывать и фильтровать данные (например, сортировать темы обсуждения по дате или фильтровать сообщения, по ключевым словам), обеспечивая предсказуемость и избежание побочных эффектов. Это особенно полезно при работе с данными, которые часто обновляются и требуют трансформации перед отображением.