**Лабораторная работа №15**

**Выбор и обоснование выбора среды разработки программы. Изучение различных стилей программирования, правил формирования листинга программы.**

**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.

**Парадигмы программирования**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название парадигмы** | **Сущность парадигмы, основные идеи, принципы, объекты.** | **Языки поддерживаю щие данную парадигму. (2-3 языка)** | **Достоинства** | **Недостатки** | **Примеры программ** |
| **1.** | [**Императивное программирование**](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) | Программный код в императивном стиле организован как последовательность отдельных команд, инструкций, описывающих логику работы программы. | Java, C, C++ | * Изучать императивный подход к разработке программ достаточно легко. * Можно писать хорошо читаемый код и отлаживать его, работая над небольшими проектами * императивная парадигма лежит в основе топовых языков программирования | * по мере роста проекта код становится тяжело поддерживать и вообще возникают проблемы с масштабированием приложений. * В больших проектах из-за ограничений архитектуры разработчики вынуждены писать избыточный код. | Фреймворки Spring, Stuts, JSP и другие позволяют писать веб-приложения: от ecommerce-проектов до крупных порталов, от образовательных платформ до правительственных ресурсов. |
| **2.** | [**Декларативное программирование**](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) | Декларативная парадигма, в отличие от императивной, описывает не последовательность инструкций, а проблему (задачу) и модель (набор выражений) для ее решения. То есть любой допустимый набор входных данных будет обработан в соответствии с моделью. | HTML + CSS, SQL | * Хорошо подходит для программирования параллельных компьютеров. * В основе лежит не какая-то машинная модель, а логика и математика. * Пригодность для формальных рассуждений | * декларативные языки программирования могут легко создать узкие места в производительности | Базы данных |
| **3.** | [**Структурное программирование**](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) | Цель структурного программирования — повысить производительность труда программистов, в том числе при разработке больших и сложных программных комплексов, сократить число ошибок, упростить отладку, модификацию и сопровождение программного обеспечения. | Pascal, Delphi | * Структурное программирование позволяет значительно сократить число вариантов построения программы по одной и той же спецификации, * В структурированных программах логически связанные операторы находятся визуально ближе, а слабо связанные — дальше, что позволяет обходиться без блок-схем и других графических форм изображения алгоритмов (по сути, сама программа является собственной блок-схемой). * Сильно упрощается процесс тестирования и отладки структурированных программ. | * невозможно полностью отказаться от меток и операторов безусловного перехода при создании сложных ПП * локальные модификации могли нарушить работоспособность всей системы. * далеко не все задачи поддаются алгоритмическому описанию и тем более алгоритмической декомпозиции, как того требует структурное программирование. | Продукция Embarcadero: Embarcadero Delphi, Embarcadero C++ Builder, Borland JBuilder 1 и 2 версии.  Администрирование и разработка баз данных: MySQL Tools, IBExpert, Open Server.  Инженерное программное обеспечение: Altium Designer, SprutCAM.  Файловые менеджеры: Total Commander, Frigate, ViewFD, FreeCommander. |
| **4.** | **Процедурное программирование** | Эта парадигма использует линейный нисходящий подход и рассматривает данные и процедуры как два разных объекта. Основываясь на концепции вызова процедуры, процедурное программирование делит программу на процедуры, которые также известны как процедуры или функции, просто содержащие последовательность шагов, которые необходимо выполнить. | Perl, Fortran, Pascal, Basic | * Процедурное программирование отлично подходит для программирования общего назначения * Исходный код переносим, поэтому его можно использовать и для другого процессора * Ход программы можно легко отследить * Благодаря методике процедурного программирования требования к памяти также сокращаются | * Код программы труднее писать, когда используется процедурное программирование * Важное значение придается операции, а не данным, что может создавать проблемы в некоторых случаях, связанных с данными * Данные открыты для всей программы, что делает их не очень безопасными | Широко используется в первую очередь для научных и инженерных вычислений. |
| **5.** | **Модульное программирование** | Модульное программирование — это организация программы как совокупности небольших независимых блоков, называемых модулями. | Pascal, Assembler, Python и Ruby | * Ускорение разработки. * Повышение надёжности. * Упрощение тестирования. * Взаимозаменяемость | Сама по себе идея использования модулей не сильно упрощает код.  Простое создание интерфейсов лишь ограничивает доступ к модулям, а для снижения сложности кода необходимо также уменьшить количество связей. Интерфейсы, хаотично ссылающиеся на другие интерфейсы, код только усложняют. | Данное программирование —общего назначения, но чаще всего применяется в веб-разработке. |
| **6.** | [**Объектно**](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)**-**[**ориентированное программирование**](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) | ООП — это подход к программированию, который распознает жизнь в том виде, в каком мы ее знаем, как совокупность объектов, которые работают в тандеме друг с другом для решения конкретной проблемы под рукой. | Java, Python, C++, PHP | * Благодаря модульности и инкапсуляции, ООП предлагает простоту управления * ООП имитирует реальный мир, облегчая понимание * Поскольку объекты являются цельными внутри себя, они могут использоваться в других программах | * Объектно-ориентированные программы, как правило, работают медленнее и занимают много памяти * Чрезмерная обобщение * Программы, созданные с использованием этой парадигмы, могут занять больше времени | ЯП данного вида программирования способен выполнять все то, что выполняет любая другая программа CGI, например, обрабатывать данные форм, генерировать динамические страницы или отсылать и принимать cookies. |
| **7.** | [**Функциональное программирование**](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) | В функциональной парадигме чистая функция всегда возвращает один и тот же результат для одних и тех же входных данных и независимо от того, когда и в какой строке кода она была запущена. | Haskell, Erlang, Scala | * Позволяет минимизировать или вовсе избавиться от мутабельности данных. * Не допускает появления побочных эффектов. * Лучше подходит для разработки масштабируемой функциональности, так как обеспечивает гораздо меньшую связность кода и высокий уровень абстракции. | * Необходимость менять входные данные при активном взаимодействии приложения с пользователями или другими приложениями заставляет разработчиков лишний подумать, стоит ли использовать эти языки. | Data Science и Big Data |
| **8.** | [**Логическое программирование**](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) | является общей платформой, объединяющей такие дисциплины как  искусственный интеллект, математическую логику и программирование. Программы в ней задаются в форме логических утверждений и правил вывода. | Пролог | * является общей платформой, объединяющей такие дисциплины как   искусственный интеллект, математическую логику и программирование | * Проблема поиска решений * При масштаб-и производит-и   увеличивается время ожидания загрузки программы | используются для создания искусственного интеллекта на основе логических моделей баз знаний и логических процедур вывода и принятия решений. |

**Вывод:**

Для реализации своего программного продукта я выбираю язык разметки HTML и стилизации CSS. HTML и CSS хороши тем, что с ними относительная простота использования, возможность уменьшения размера сайта, множество дополнительных возможностей по стилизации текста и другого контента. Для фронтэнд разработки – это максимально удобные языки. Также он хорошо подходит программирования параллельных компьютеров, в основе лежит не какая-то машинная модель, а логика и математика, пригодность для формальных рассуждений.