**Конспект по теме «Этап реализации»**

**Язык** **программирования** — формальный язык, предназначенный для записи компьютерных программ.

**Синтаксис** **языка** **программирования** - это правила составления предложений **языка** из отдельных предложений.

**Алфавит языка программирования** — это все символы или комбинации символов, которые используются при программировании на этом **языке**.

**Семантика** в **программировании** — дисциплина, изучающая формализации значений конструкций языков программирования посредством построения их формальных математических моделей.

У каждого языка программирования есть специальный документ, который называется **стандартом языка программирования**.

Классификация языков программирования:

* 1. Процедурные языки
  2. Языки программирования низкого уровня
  3. Языки программирования высокого уровня
  4. Объектно-ориентированные языки
  5. Декларативные языки программирования
  6. Функциональные языки программирования
  7. Логические языки программирования
  8. Языки сценариев (скрипты)
  9. Языки, ориентированные на данные

Критерии выбора языка программирования:

1. **Скорость работы конечного продукта.**  
   Для данных целей хорошо подойдут компилируемые языки: *ассемблер, С/С++, фортран* и т.д. После сборки программа не требует ничего лишнего и содержит в себе машинные команды, которые выполняются без лишних задержек.
2. **Объем занимаемой оперативной памяти.**  
   Данное требование появляется, когда программа разрабатывается для встраиваемых систем, мобильных платформ, микроконтроллеров и так далее. В данных случаях, чем меньше памяти расходует программа на данном языке – тем лучше. К таким языкам, опять же, относятся *ассемблер, С/С++, Objective-C* и другие.
3. **Скорость разработки программы.**  
   На данных языках время разработки может существенно сокращаться из-за обилия сторонних библиотек, максимально «очеловеченного» синтаксиса, и подобных вещей. Это, например, *Java, Flash* и подобные. Скорость выполнения программ, написанных на данных языках страдает, причем порой весьма ощутимо.
4. **Кроссплатформенность.**  
   Кроссплатформенность – возможность работы программы на различных платформах, в различных ОС с минимальными изменениями. В этой сфере можно выделить такие языки: *Java, C#,Flash,C++* с различными библиотеками и другие, менее используемые, языки.  
   В этот раздел так же можно включить интерпретируемые, скриптовые языки – для их работы нужно наличие интерпретатора языка в системе. Данные языки очень удобны в плане разработки, но достаточно медлительны.
5. **Скорость внесения изменений, скорость тестирования**  
   Проект стремительно развивается, в него постоянно вносятся изменения, порой немало? Тогда выбор должен падать на высокоуровневые языки, где любой функциональный блок можно быстро переписать.

**Парадигма** **программирования** — это совокупность идей и понятий, определяющих стиль написания компьютерных программ. Это способ [концептуализации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BB%D0%B8%D0%BD%D0%B3%D0%B2%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)), определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

**Процедурное программирование** – это программирование при котором, при котором программа представляет собой последовательность операторов. Используется в языках высокого уровня Basic, Fortran и др.

**Функциональное программирование** – это программирование, при котором программа представляет собой последовательность вызовов функций. Используется в языках Lisp и др.

**Логическое программирование** – это программирование, при котором программа представляет собой совокупность определения соотношений между объектами. Используется в языках Prolog и др.

**Объектно-ориентированное программирование** – это программирование, при котором основой программы является объект, представляющий собой совокупность данных и правил их преобразования. Используется в языках Turbo-Pascal, C++ и др.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название парадигмы** | **Сущность**  **парадигмы,**  **основные**  **идеи,**  **принципы,**  **объекты.** | **Языки**  **поддерживающие данную**  **парадигму.**  **(2-3 языка)** | **Достоинства** | **Недостатки** |
| **1** | Императивное  программирование | В исходном коде записываются «приказы» команды, а не классы  Все инструкции должны выполняться последовательно, один за другим.  После выполнения инструкций данные могут записываться в память и считываться памяти. | Ассамблер, fortran, algo | эффективная реализация | Нет возможности перескакивать с одного участка кода на другой |
| **2** | Декларативное  программирование | Декларативная программа состоит из ограничений и правил, из которых компьютер генерирует способ получения результата | SQL, MySQL | позволяет писать более безопасный и поддерживаемый код, который легко параллелится. А компиляторы декларативных языков имеют больше возможностей при оптимизации программ. | Сложно в понимании |
| **3** | Структурное  программирование | представление программы в виде иерархической структуры блоков. | C, Pascal, Basic | позволяет сократить число вариантов построения программы по одной и той же спецификации  логически связанные операторы находятся визуально ближе, а слабо связанные — дальше  Сильно упрощается процесс тестирования и отладки структурированных программ. | работа со структурными данными практически не возможна без использования скриптов и знания как эти скрипты функционируют |
| **4** | Процедурное  программирование | процедура может иметь несколько точек выхода (return в C-подобных языках), несколько точек входа, иметь аргументы, возвращать значение как результат своего выполнения, быть перегруженной по количеству или типу параметров и много чего еще. | C, C++, Pascal | отлично подходит для программирования общего назначения  Закодированная простота наряду с простотой реализации компиляторов и интерпретаторов  Исходный код переносим  Код может быть повторно использован в разных частях программы, без необходимости копировать его  Благодаря методике процедурного программирования требования к памяти также сокращаются  Ход программы можно легко отследить | Код программы труднее писать  Процедурный код часто не может быть использован повторно, что может привести к необходимости воссоздания кода, если это необходимо для использования в другом приложении.  Сложно общаться с реальными объектами  Данные открыты для всей программы, что делает их не очень безопасными |
| **5** | Модульное  программирование | Программа описанная в стиле модульного программирования — это набор модулей. Что внутри, классы, императивный код или чистые функции — не важно. | Haskell, Pascal | Повышение удобства сопровождения кода.  Совместная работа нескольких человек не мешает друг другу. | невозможно загрузить модули по запросу. |
| **6** | Объектно-  ориентированное  программирование | основанно на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования | Java, Python, C# | **Параллельная разработка, Модульность, Безопасность. Возможность повторного использования** | Низкая **производительность**  **Требуется больше планирования** |
| **7** | Функциональное  программирование | Программы создаются путем последовательного применения функций, а не инструкций | Haskell, F# | Легкая отладка, Параллельное программирование, Модульность, Отложенное вычисление | плохо подходит для алгоритмов, основанных на графах из-за сравнительно более медленной работы программы |
| **8** | Логическое  программирование | Подход к программированию, при  котором программа задаѐтся совокупностью правил без явного указания  последовательности их применения. | Prolog, Delta Prolog | Компактность кода, Легкость понимания, Эффективный метод вычислений - рекурсия | Невозможность создания комплексных задач |

**Грамотный выбор идентификаторов:**

* Поднимайтесь на один уровень абстракции выше тела элемента (за исключением имён классов).
* Имя класса должно описывать его обязанность.
* Уважайте принцип единственной обязанности (одно из SOLID-правил).
* Разбивайте задачу на более мелкие подзадачи.

**Стандарт** **оформления** **кода** — набор правил и соглашений, используемых при написании исходного кода на некотором языке программирования. Наличие общего стиля программирования облегчает понимание и поддержание исходного кода, написанного более чем одним программистом, а также упрощает взаимодействие нескольких человек при разработке программного обеспечения.

**Венгерская** **нотация** в программировании — соглашение об именовании переменных, констант и прочих идентификаторов в коде программ.

**Верблюжья** нотация — практика написание текстов, которая придерживается принципа игнорирования пробелов и знаков препинания.